

トランسفォーマーによるエンドツーエンドの物体検出

ニコラス・カリオン^{*}、フランシスコ・マサ^{*}、ガブリエル・シンネフ、ニコラ・ウスニエ、アレクサンダー・キロフ、セルゲイ・ザゴルイコ

Facebook AI

概要。物体検出を直接集合予測問題として捉える新しい手法を紹介する。我々のアプローチは検出パイプラインを合理化し、タスクに関する我々の事前知識を明示的にエンコードする非最大抑制手順やアンカー生成のような多くの手作業で設計されたコンポーネントの必要性を効果的に排除する。DEtection TRansformer (DETR) と呼ばれる新しいフレームワークの主な構成要素は、二分割マッチングによって一意な予測を強制するセットベースのグローバル損失と、変換エンコーダ・デコーダアーキテクチャである。学習されたオブジェクトクエリの固定された小さなセットが与えられたとき、DETRはオブジェクトとグローバルな画像コンテキストの関係を推論し、予測の最終セットを直接並列に出力する。新しいモデルは概念的に単純であり、他の多くの最新の検出器とは異なり、特殊なライブラリを必要としない。DETRは、難易度の高いCOCO物体検出データセットにおいて、確立され、高度に最適化されたFaster RCNNベースラインと同等の精度と実行時間性能を実証した。さらに、DETRは汎光分割を統一的に生成するために容易に一般化することができる。競合するベースラインを大幅に上回ることを示す。学習コードと事前学習済みモデルは <https://github.com/facebookresearch/detr> で公開されている。

1 はじめに

物体検出の目的は、関心のある各物体のバウンディングボックスとカテゴリラベルのセットを予測することである。現代の検出器は、大規模な提案セット[37, 5]、アンカー[23]、またはウィンドウセンター[53, 46]上の代理回帰と分類問題を定義することによって、間接的な方法でこのセット予測タスクに対処している。彼らの性能は、重複に近い予測を崩壊させる後処理ステップ、アンカーセットの設計、アンカーにターゲットボックスを割り当てるヒューリスティックスによって大きく影響される[52]。これらのパイプラインを単純化するために、サロゲートタスクをバイパスする直接集合予測アプローチを提案する。このエンドツーエンドの哲学は、機械翻訳や音声認識のような複雑な構造化予測タスクにおいて大きな進歩をもたらしたが、物体検出においてはまだ進歩していない。以前の試み[43, 16, 4, 39]は、他の形式の事前知識を追加するか、困難なベンチマークにおいて強力なベースラインと競合することが証明されていない。本稿は、このギャップを埋めることを目的とする。

* 均等な貢献

End-to-End Object Detection with Transformers

Nicolas Carion*, Francisco Massa*, Gabriel Synnaeve, Nicolas Usunier,
Alexander Kirillov, and Sergey Zagoruyko

Facebook AI

Abstract. We present a new method that views object detection as a direct set prediction problem. Our approach streamlines the detection pipeline, effectively removing the need for many hand-designed components like a non-maximum suppression procedure or anchor generation that explicitly encode our prior knowledge about the task. The main ingredients of the new framework, called DEtection TRansformer or DETR, are a set-based global loss that forces unique predictions via bipartite matching, and a transformer encoder-decoder architecture. Given a fixed small set of learned object queries, DETR reasons about the relations of the objects and the global image context to directly output the final set of predictions in parallel. The new model is conceptually simple and does not require a specialized library, unlike many other modern detectors. DETR demonstrates accuracy and run-time performance on par with the well-established and highly-optimized Faster R-CNN baseline on the challenging COCO object detection dataset. Moreover, DETR can be easily generalized to produce panoptic segmentation in a unified manner. We show that it significantly outperforms competitive baselines. Training code and pretrained models are available at <https://github.com/facebookresearch/detr>.

1 Introduction

The goal of object detection is to predict a set of bounding boxes and category labels for each object of interest. Modern detectors address this set prediction task in an indirect way, by defining surrogate regression and classification problems on a large set of proposals [37,5], anchors [23], or window centers [53,46]. Their performances are significantly influenced by postprocessing steps to collapse near-duplicate predictions, by the design of the anchor sets and by the heuristics that assign target boxes to anchors [52]. To simplify these pipelines, we propose a direct set prediction approach to bypass the surrogate tasks. This end-to-end philosophy has led to significant advances in complex structured prediction tasks such as machine translation or speech recognition, but not yet in object detection: previous attempts [43,16,4,39] either add other forms of prior knowledge, or have not proven to be competitive with strong baselines on challenging benchmarks. This paper aims to bridge this gap.

* Equal contribution

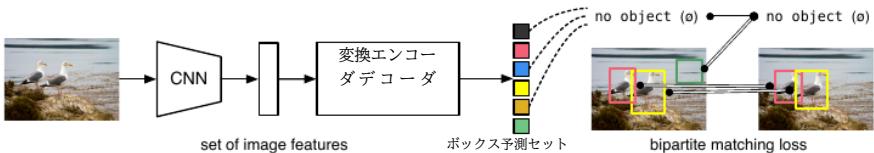


図1: DETRは、一般的なCNNと変換アーキテクチャを組み合わせることで、最終的な検出セットを(並列に)直接予測する。学習中、二分割マッチングは予測値をグランドトゥルースボックスで一意に割り当てる。一致しない予測は、「オブジェクトなし」(Δ)クラス予測をもたらすはずである。

物体検出を直接集合予測問題とみなすことで、学習パイプラインを合理化する。我々は、シーケンス予測のための一般的なアーキテクチャである、変換器[47]に基づくエンコーダ・デコーダアーキテクチャを採用する。トランسفォーマーの自己アテンションメカニズムは、シーケンス内の要素間のすべてのペアワイズ相互作用を明示的にモデル化し、これらのアーキテクチャは、重複予測の削除のようなセット予測の特定の制約に特に適している。

我々のDEtection TRansformer (DETR, 図1参照)は、一度に全てのオブジェクトを予測し、予測されたオブジェクトとグランドトゥルースオブジェクトの間の二分割マッチングを実行するセット損失関数でエンドツーエンドで学習される。DETRは、空間アンカーや非最大抑制のような、事前知識をエンコードする複数の手作業で設計されたコンポーネントを削除することで、検出パイプラインを簡素化する。既存の多くの検出方法とは異なり、DETRはカスタマイズされたレイヤーを必要としないため、標準的なCNNと変換器クラスを含むどのようなフレームワークでも簡単に再現できる¹。

直接集合予測に関する多くの先行研究と比較して、DETRの主な特徴は、二分割マッチング損失と(非自己回帰)並列復号化を持つ変換器[29, 12, 10, 8]の結合である。一方、先行研究では、RNNを用いた自己回帰復号に着目している[43, 41, 30, 36, 42]。我々のマッチング損失関数は、予測を一意に基底真理オブジェクトに割り当て、予測オブジェクトの並べ替えに対して不变であるため、並列にそれらを放出することができる。

最も人気のある物体検出データセットの1つであるCOCO [24]で、非常に競争力のあるFaster R-CNNベースライン[37]に対してDETRを評価する。より高速なRCNNは何度も設計の繰り返しが行われ、その性能は元の出版物よりも大幅に改善された。我々の実験は、我々の新しいモデルが同等の性能を達成することを示している。より正確には、DETRは大きな物体に対して有意に優れた性能を示し、この結果は変換器の非局所的な計算によって可能になったと考えられる。しかし、小さな物体では性能が低下する。今後の研究により、Faster R-CNNに対するFPN [22]の開発と同じ方法で、この点を改善することが期待される。

DETRの学習設定は、標準的な物体検出器とは複数の点で異なる。この新しいモデルは、余分な学習スケジュールを必要とし、

¹ In our work we use standard implementations of Transformers [47] and ResNet [15] backbones from standard deep learning libraries.

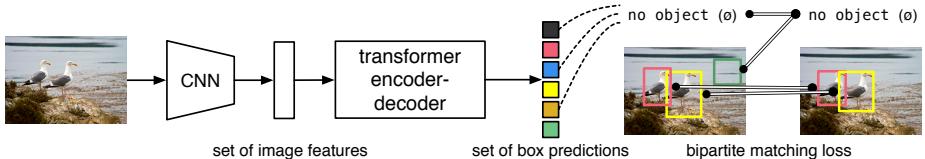


Fig. 1: DETR directly predicts (in parallel) the final set of detections by combining a common CNN with a transformer architecture. During training, bipartite matching uniquely assigns predictions with ground truth boxes. Prediction with no match should yield a “`no object`” (\emptyset) class prediction.

We streamline the training pipeline by viewing object detection as a direct set prediction problem. We adopt an encoder-decoder architecture based on transformers [47], a popular architecture for sequence prediction. The self-attention mechanisms of transformers, which explicitly model all pairwise interactions between elements in a sequence, make these architectures particularly suitable for specific constraints of set prediction such as removing duplicate predictions.

Our DEtection TRansformer (DETR, see Figure 1) predicts all objects at once, and is trained end-to-end with a set loss function which performs bipartite matching between predicted and ground-truth objects. DETR simplifies the detection pipeline by dropping multiple hand-designed components that encode prior knowledge, like spatial anchors or non-maximal suppression. Unlike most existing detection methods, DETR doesn’t require any customized layers, and thus can be reproduced easily in any framework that contains standard CNN and transformer classes.¹

Compared to most previous work on direct set prediction, the main features of DETR are the conjunction of the bipartite matching loss and transformers with (non-autoregressive) parallel decoding [29, 12, 10, 8]. In contrast, previous work focused on autoregressive decoding with RNNs [43, 41, 30, 36, 42]. Our matching loss function uniquely assigns a prediction to a ground truth object, and is invariant to a permutation of predicted objects, so we can emit them in parallel.

We evaluate DETR on one of the most popular object detection datasets, COCO [24], against a very competitive Faster R-CNN baseline [37]. Faster R-CNN has undergone many design iterations and its performance was greatly improved since the original publication. Our experiments show that our new model achieves comparable performances. More precisely, DETR demonstrates significantly better performance on large objects, a result likely enabled by the non-local computations of the transformer. It obtains, however, lower performances on small objects. We expect that future work will improve this aspect in the same way the development of FPN [22] did for Faster R-CNN.

Training settings for DETR differ from standard object detectors in multiple ways. The new model requires extra-long training schedule and benefits

¹ In our work we use standard implementations of Transformers [47] and ResNet [15] backbones from standard deep learning libraries.

変換器の補助的な復号損失の恩恵を受ける。実証された性能のために、どのような要素が重要であるかを徹底的に探る。

DETRの設計理念は、より複雑なタスクにも容易に拡張できる。我々の実験では、最近人気を博している困難なピクセルレベルの認識タスクであるPanoptic Segmentation [19]において、事前に訓練されたDETRの上に訓練された単純なセグメンテーションヘッドが、競争力のあるベースラインを上回ることを示す。

2 Related work

我々の研究は、集合予測のための二分割マッチング損失、変換器に基づくエンコーダ・デコーダーアーキテクチャ、並列デコーダ、物体検出法などのいくつかの領域における先行研究を基礎としている。

2.1 集合予測

集合を直接予測する正統的な深層学習モデルは存在しない。基本的な集合予測タスクはマルチラベル分類であり(コンピュータビジョンの文脈での参考文献は[40, 33]など参照)、ベースラインアプローチであるone-vs-restは、要素間に基礎構造(すなわち、ほぼ同一のボックス)がある検出などの問題には適用されない。これらのタスクの最初の困難は、重複に近い状態を避けることである。現在の検出器の多くは、この問題に対処するために非最大抑制などの後処理を使用しているが、直接集合予測は後処理を行わない。冗長性を避けるために、全ての予測要素間の相互作用をモデル化するグローバル推論スキームが必要である。定サイズ集合予測では、密な完全連結ネットワーク[9]で十分であるが、コストがかかる。一般的なアプローチは、リカレントニューラルネットワーク[48]のような自己回帰シーケンスモデルを使用することである。すべての場合において、損失関数は予測値の並べ替えによって不变であるべきである。通常の解決策は、ハンガリーアルゴリズム[20]に基づいて損失を設計し、グランドトゥルースと予測の間の二分割マッチングを見つけることである。これは順列不变性を強制し、各ターゲット要素が一意にマッチすることを保証する。我々は二分割マッチング損失アプローチに従う。しかし、多くの先行研究とは対照的に、我々は自己回帰モデルから離れ、並列復号化を持つ変換器を使用する。

2.2 トランスフォーマーと並列復号化

トランスフォーマーはVaswaniらによって紹介された。[47]を機械翻訳のための新しい注意ベースのビルディングブロックとして提案する。注意メカニズム[2]は、入力シーケンス全体から情報を集約するニューラルネットワーク層である。Transformerは自己注意層を導入し、Non-Local Neural Networks [49]と同様に、シーケンスの各要素をスキャンし、シーケンス全体の情報を集約して更新する。注意に基づくモデルの主な利点の1つは、大域的な計算と完全なメモリであり、長いシーケンスに対するRNNよりも適している。

from auxiliary decoding losses in the transformer. We thoroughly explore what components are crucial for the demonstrated performance.

The design ethos of DETR easily extend to more complex tasks. In our experiments, we show that a simple segmentation head trained on top of a pre-trained DETR outperforms competitive baselines on Panoptic Segmentation [19], a challenging pixel-level recognition task that has recently gained popularity.

2 Related work

Our work build on prior work in several domains: bipartite matching losses for set prediction, encoder-decoder architectures based on the transformer, parallel decoding, and object detection methods.

2.1 Set Prediction

There is no canonical deep learning model to directly predict sets. The basic set prediction task is multilabel classification (see e.g., [40,33] for references in the context of computer vision) for which the baseline approach, one-vs-rest, does not apply to problems such as detection where there is an underlying structure between elements (i.e., near-identical boxes). The first difficulty in these tasks is to avoid near-duplicates. Most current detectors use postprocessings such as non-maximal suppression to address this issue, but direct set prediction are postprocessing-free. They need global inference schemes that model interactions between all predicted elements to avoid redundancy. For constant-size set prediction, dense fully connected networks [9] are sufficient but costly. A general approach is to use auto-regressive sequence models such as recurrent neural networks [48]. In all cases, the loss function should be invariant by a permutation of the predictions. The usual solution is to design a loss based on the Hungarian algorithm [20], to find a bipartite matching between ground-truth and prediction. This enforces permutation-invariance, and guarantees that each target element has a unique match. We follow the bipartite matching loss approach. In contrast to most prior work however, we step away from autoregressive models and use transformers with parallel decoding, which we describe below.

2.2 Transformers and Parallel Decoding

Transformers were introduced by Vaswani *et al.* [47] as a new attention-based building block for machine translation. Attention mechanisms [2] are neural network layers that aggregate information from the entire input sequence. Transformers introduced self-attention layers, which, similarly to Non-Local Neural Networks [49], scan through each element of a sequence and update it by aggregating information from the whole sequence. One of the main advantages of attention-based models is their global computations and perfect memory, which makes them more suitable than RNNs on long sequences. Transformers are now

自然言語処理、音声処理、コンピュータビジョンの多くの問題で、トランスフォーマーがRNNに取って代わりつつある[8, 27, 45, 34, 31]。トランスフォーマーは、初期のsequence-to-sequenceモデル[44]に従い、出力トークンを1つずつ生成する自己回帰モデルで最初に使用された。しかし、法外な推論コスト(出力長に比例し、バッチ処理が難しい)により、音声[29]、機械翻訳[12, 10]、単語表現学習[8]、そして最近では音声認識[6]の領域で、並列シーケンス生成の発展が進んでいる。また、計算コストと集合予測に必要な大域的計算を実行する能力との間の適切なトレードオフのために、変換器と並列復号化を組み合わせる。

2.3 物体検出

最新の物体検出手法の多くは、いくつかの初期推測に対して予測を行う。二段検出器[37, 5]は提案に対してボックスを予測し、一段検出法はアンカー[23]や可能なオブジェクト中心のグリッド[53, 46]に対して予測を行う。最近の研究[52]は、これらのシステムの最終的な性能が、これらの初期推測の正確な設定方法に大きく依存することを示している。我々のモデルでは、この手作業によるプロセスを削除し、アンカーではなく入力画像に対して絶対的なボックス予測で検出の集合を直接予測することで、検出プロセスを合理化することができる。

集合ベースの損失。いくつかの物体検出器[9, 25, 35]は二分割マッチング損失を用いている。しかし、これらの初期のディープラーニングモデルでは、異なる予測間の関係は畳み込み層または完全連結層のみでモデル化されており、手作業で設計されたNMS後処理によって性能を向上させることができる。より最近の検出器[37, 23, 53]は、NMSとともに、グランドトゥルースと予測値の間の非一意な割り当てルールを使用する。

学習可能なNMS手法[16, 4]と関係ネットワーク[17]は、異なる予測間の関係を注意で明示的にモデル化する。直接設定された損失を使用するため、後処理ステップを必要としない。しかし、これらの手法は、提案ボックス座標のような手作業で作成された追加のコンテキスト特徴を採用し、検出間の関係を効率的にモデル化する一方で、モデルにエンコードされた事前知識を削減するソリューションを探す。

リカレント検出器。我々のアプローチに最も近いのは、物体検出[43]とインスタンス分割[41, 30, 36, 42]のためのエンドツーエンドのセット予測である。我々と同様に、彼らはCNNの活性化に基づくエンコーダ・デコーダのアーキテクチャで二分割マッチング損失を使い、直接バウンディングボックスのセットを生成する。しかし、これらのアプローチは小規模なデータセットでのみ評価され、最新のベースラインとは比較されていない。特に、自己回帰モデル(より正確にはRNN)に基づいているため、並列復号化を伴う最近の変換器を活用していない。

3 DETRモデル

検出における直接集合予測には、次の2つの要素が不可欠である:(1)予測されたボックスとグランドトゥルースボックスの間の一意なマッチングを強制する集合予測損失、

replacing RNNs in many problems in natural language processing, speech processing and computer vision [8,27,45,34,31].

Transformers were first used in auto-regressive models, following early sequence-to-sequence models [44], generating output tokens one by one. However, the prohibitive inference cost (proportional to output length, and hard to batch) lead to the development of parallel sequence generation, in the domains of audio [29], machine translation [12,10], word representation learning [8], and more recently speech recognition [6]. We also combine transformers and parallel decoding for their suitable trade-off between computational cost and the ability to perform the global computations required for set prediction.

2.3 Object detection

Most modern object detection methods make predictions relative to some initial guesses. Two-stage detectors [37,5] predict boxes w.r.t. proposals, whereas single-stage methods make predictions w.r.t. anchors [23] or a grid of possible object centers [53,46]. Recent work [52] demonstrate that the final performance of these systems heavily depends on the exact way these initial guesses are set. In our model we are able to remove this hand-crafted process and streamline the detection process by directly predicting the set of detections with absolute box prediction w.r.t. the input image rather than an anchor.

Set-based loss. Several object detectors [9,25,35] used the bipartite matching loss. However, in these early deep learning models, the relation between different prediction was modeled with convolutional or fully-connected layers only and a hand-designed NMS post-processing can improve their performance. More recent detectors [37,23,53] use non-unique assignment rules between ground truth and predictions together with an NMS.

Learnable NMS methods [16,4] and relation networks [17] explicitly model relations between different predictions with attention. Using direct set losses, they do not require any post-processing steps. However, these methods employ additional hand-crafted context features like proposal box coordinates to model relations between detections efficiently, while we look for solutions that reduce the prior knowledge encoded in the model.

Recurrent detectors. Closest to our approach are end-to-end set predictions for object detection [43] and instance segmentation [41,30,36,42]. Similarly to us, they use bipartite-matching losses with encoder-decoder architectures based on CNN activations to directly produce a set of bounding boxes. These approaches, however, were only evaluated on small datasets and not against modern baselines. In particular, they are based on autoregressive models (more precisely RNNs), so they do not leverage the recent transformers with parallel decoding.

3 The DETR model

Two ingredients are essential for direct set predictions in detection: (1) a set prediction loss that forces unique matching between predicted and ground truth

(2) オブジェクトの集合を(1回のパスで)予測し、それらの関係をモデル化するアーキテクチャ。我々のアーキテクチャを図2に詳しく説明する。

3.1 物体検出セット予測損失

DETRは、デコーダを1回通過するだけで、 N 個の予測値の固定サイズのセットを推論する。ここで、 N は画像中の典型的なオブジェクトの数よりもかなり大きくなるように設定される。学習の主な困難の一つは、予測されたオブジェクト(クラス、位置、サイズ)をグランドトゥルースに関してスコアリングすることである。我々の損失は、予測されたオブジェクトとグランドトゥルースのオブジェクトの間の最適な二分割マッチングを生成し、次にオブジェクト固有の(パウンディングボックス)損失を最適化する。

ここで、オブジェクトの基底真理集合を y 、 N 個の予測集合を $\hat{y} = \{\hat{y}_i\}_{i=1}^N$ とする。 N が画像中のオブジェクトの数より大きいと仮定すると、 y も ϕ でパディングされたサイズ N の集合(オブジェクトなし)と考える。この2つの集合の2分割マッチングを見つけるために、最もコストの低い N 個の要素 $\sigma \in S_N$ の並べ替えを探索する：

$$\hat{\sigma} = \arg \min_{\sigma \in S_N} \sum_i^N \mathcal{L}_{\text{match}}(y_i, \hat{y}_{\sigma(i)}), \quad (1)$$

ここで、 $\mathcal{L}_{\text{match}}(y_i, \hat{y}_{\sigma(i)})$ は、グランドトゥルース y_i とインデックス $\sigma(i)$ を持つ予測との間のペアワイズマッチングコストである。この最適割り当ては、先行研究(例えば[43])に従い、ハンガリーアルゴリズムで効率的に計算される。

マッチングコストは、クラス予測と、予測されたボックスとグランドトゥルースボックスの類似性の両方を考慮する。ここで、 c_i はターゲットクラスラベル(ϕ)であり、 $b_i \in [0, 1]^4$ はグランドトゥルースボックスの中心座標と画像サイズに対する高さと幅を定義するベクトルである。インデックス $\sigma(i)$ を持つ予測に対して、クラス c_i の確率を $p_{\sigma(i)}(c_i)$ と定義し、予測ボックスを $\hat{b}_{\sigma(i)}$ と定義する。これらの表記を用いて、 $\mathcal{L}_{\text{match}}(y_i, \hat{y}_{\sigma(i)})$ を $-1 \{c_i = \phi\} \{p\} \sigma(i) (c_i) + 1 \{c_i = \phi\} \{b_i\} \hat{b}_{\sigma(i)}$ と定義する。

このマッチングを見つける手順は、提案[37]やアンカー[22]を最新の検出器のグランドトゥルースオブジェクトにマッチングさせるために使用される発見的割り当てルールと同じ役割を果たす。主な違いは、重複のない直接集合予測のために、一对一のマッチングを見つける必要があることである。

第2ステップは、前のステップでマッチしたすべてのペアの損失関数、ハンガリー損失を計算することである。我々は、一般的な物体検出器の損失と同様に、クラス予測のための負の対数尤度と後で定義されるボックス損失の線形結合を定義する：

$$\mathcal{L}_{\text{Hungarian}}(y, \hat{y}) = \sum_{i=1}^N \left[-\log \hat{p}_{\hat{\sigma}(i)}(c_i) + \mathbb{1}_{\{c_i \neq \phi\}} \mathcal{L}_{\text{box}}(b_i, \hat{b}_{\hat{\sigma}(i)}) \right], \quad (2)$$

ここで、 $\hat{\sigma}$ は最初のステップ(1)で計算された最適割り当てである。実際には、クラスの不均衡を考慮し、 $c_i = \phi$ のときの対数確率項を10倍ダウンウェイトする。

boxes; (2) an architecture that predicts (in a single pass) a set of objects and models their relation. We describe our architecture in detail in Figure 2.

3.1 Object detection set prediction loss

DETR infers a fixed-size set of N predictions, in a single pass through the decoder, where N is set to be significantly larger than the typical number of objects in an image. One of the main difficulties of training is to score predicted objects (class, position, size) with respect to the ground truth. Our loss produces an optimal bipartite matching between predicted and ground truth objects, and then optimize object-specific (bounding box) losses.

Let us denote by y the ground truth set of objects, and $\hat{y} = \{\hat{y}_i\}_{i=1}^N$ the set of N predictions. Assuming N is larger than the number of objects in the image, we consider y also as a set of size N padded with \emptyset (no object). To find a bipartite matching between these two sets we search for a permutation of N elements $\sigma \in \mathfrak{S}_N$ with the lowest cost:

$$\hat{\sigma} = \arg \min_{\sigma \in \mathfrak{S}_N} \sum_i^N \mathcal{L}_{\text{match}}(y_i, \hat{y}_{\sigma(i)}), \quad (1)$$

where $\mathcal{L}_{\text{match}}(y_i, \hat{y}_{\sigma(i)})$ is a pair-wise *matching cost* between ground truth y_i and a prediction with index $\sigma(i)$. This optimal assignment is computed efficiently with the Hungarian algorithm, following prior work (*e.g.* [43]).

The matching cost takes into account both the class prediction and the similarity of predicted and ground truth boxes. Each element i of the ground truth set can be seen as a $y_i = (c_i, b_i)$ where c_i is the target class label (which may be \emptyset) and $b_i \in [0, 1]^4$ is a vector that defines ground truth box center coordinates and its height and width relative to the image size. For the prediction with index $\sigma(i)$ we define probability of class c_i as $\hat{p}_{\sigma(i)}(c_i)$ and the predicted box as $\hat{b}_{\sigma(i)}$. With these notations we define $\mathcal{L}_{\text{match}}(y_i, \hat{y}_{\sigma(i)})$ as $-\mathbb{1}_{\{c_i \neq \emptyset\}} \hat{p}_{\sigma(i)}(c_i) + \mathbb{1}_{\{c_i \neq \emptyset\}} \mathcal{L}_{\text{box}}(b_i, \hat{b}_{\sigma(i)})$.

This procedure of finding matching plays the same role as the heuristic assignment rules used to match proposal [37] or anchors [22] to ground truth objects in modern detectors. The main difference is that we need to find one-to-one matching for direct set prediction without duplicates.

The second step is to compute the loss function, the *Hungarian loss* for all pairs matched in the previous step. We define the loss similarly to the losses of common object detectors, *i.e.* a linear combination of a negative log-likelihood for class prediction and a box loss defined later:

$$\mathcal{L}_{\text{Hungarian}}(y, \hat{y}) = \sum_{i=1}^N \left[-\log \hat{p}_{\hat{\sigma}(i)}(c_i) + \mathbb{1}_{\{c_i \neq \emptyset\}} \mathcal{L}_{\text{box}}(b_i, \hat{b}_{\hat{\sigma}(i)}) \right], \quad (2)$$

where $\hat{\sigma}$ is the optimal assignment computed in the first step (1). In practice, we down-weight the log-probability term when $c_i = \emptyset$ by a factor 10 to account for

これは、Faster R-CNNの学習手順が、サブサンプリングによって正/負の提案のバランスをとる方法と類似している[37]。物体と $*$ のマッチングコストは予測に依存しないことに注意。マッチングコストでは対数確率の代わりに確率 $\hat{\sigma}^*(i)$ (c_i) を用いる。これにより、クラス予測項は L_{box} (-, -)(後述)に適合し、より良い経験的性能が観測された。

バウンディングボックス損失。マッチングコストとハンガリー損失の第2部分は、バウンディングボックスをスコアリングする L_{box} (-)である。いくつかの初期推測に対してボックス予測を行う多くの検出器とは異なり、我々はボックス予測を直接行う。このようなアプローチは実装を単純化するが、損失の相対的なスケーリングに問題がある。最もよく使われる ℓ_1 損失は、相対誤差が同程度であっても、小さい箱と大きい箱でスケールが異なる。この問題を軽減するために、 ℓ_1 損失と一般化IoU損失[38] L_{iou} (-, -)の線形結合を用いる。全体として、ボックス損失は $L_{box}(b_i, \hat{b}_{\sigma(i)})$ であり、 $\lambda_{iou} L_{iou}(b_i, \hat{b}_{\sigma(i)}) + \lambda_{L1} \|b_i - \hat{b}_{\sigma(i)}\|_1$ ここで λ_{iou} , $\lambda_{L1} \in \mathbb{R}$ はハイパーパラメータである。これら2つの損失は、バッチ内のオブジェクトの数で正規化される。

3.2 DETRアーキテクチャ

全体的なDETRアーキテクチャは驚くほど単純で、図2に描かれている。コンパクトな特徴表現を抽出するCNNバックボーン、エンコーダ・デコーダ変換器、そして最終的な検出予測を行う単純なフィードフォワードネットワーク(FFN)である。

多くの最新の検出器とは異なり、DETRは、共通のCNNバックボーンと、わずか数百行の変換器アーキテクチャ実装を提供する、任意の深層学習フレームワークで実装することができる。DETRの推論コードはPyTorch [32]で50行未満で実装できる。本手法のシンプルさが、新たな研究者の検出コミュニティへの誘致となることを期待している。

バックボーン。初期画像 $x_{img} \in \mathbb{R}^{3 \times H_0 \times W_0}$ (3色チャンネル²)から出発し、従来のCNNバックボーンは低解像度の活性化マップ $f \in \mathbb{R}^{C \times H \times W}$ を生成する。典型的な値は、 $C = 2048$, $H, W = \frac{H_0}{32}, \frac{W_0}{32}$ である。

トランスフォーマーエンコーダ。まず、 1×1 畳み込みにより、高レベル活性化マップ f のチャンネル次元を C からより小さな次元 d に縮小する。新しい特徴マップ $z_0 \in \mathbb{R}^{d \times H \times W}$ を作成する。エンコーダはシーケンスを入力として期待するので、 z_0 の空間次元を1次元に畳み込み、 $d \times HW$ の特徴マップを得る。各エンコーダ層は標準的なアーキテクチャを持ち、マルチヘッド自己アテンションモジュールとフィードフォワードネットワーク(FFN)から構成される。変換器アーキテクチャは順列不变であるため、各注意層の入力に追加される固定位置エンコーディング[31, 3]で補足する。アーキテクチャの詳細な定義については、[47]に記述されているものを踏襲して補足資料に譲る。

² The input images are batched together, applying 0-padding adequately to ensure they all have the same dimensions (H_0, W_0) as the largest image of the batch.

class imbalance. This is analogous to how Faster R-CNN training procedure balances positive/negative proposals by subsampling [37]. Notice that the matching cost between an object and \emptyset doesn't depend on the prediction, which means that in that case the cost is a constant. In the matching cost we use probabilities $\hat{p}_{\sigma(i)}(c_i)$ instead of log-probabilities. This makes the class prediction term commensurable to $\mathcal{L}_{\text{box}}(\cdot, \cdot)$ (described below), and we observed better empirical performances.

Bounding box loss. The second part of the matching cost and the Hungarian loss is $\mathcal{L}_{\text{box}}(\cdot)$ that scores the bounding boxes. Unlike many detectors that do box predictions as a Δ w.r.t. some initial guesses, we make box predictions directly. While such approach simplify the implementation it poses an issue with relative scaling of the loss. The most commonly-used ℓ_1 loss will have different scales for small and large boxes even if their relative errors are similar. To mitigate this issue we use a linear combination of the ℓ_1 loss and the generalized IoU loss [38] $\mathcal{L}_{\text{iou}}(\cdot, \cdot)$ that is scale-invariant. Overall, our box loss is $\mathcal{L}_{\text{box}}(b_i, \hat{b}_{\sigma(i)})$ defined as $\lambda_{\text{iou}}\mathcal{L}_{\text{iou}}(b_i, \hat{b}_{\sigma(i)}) + \lambda_{\text{L1}}\|b_i - \hat{b}_{\sigma(i)}\|_1$ where $\lambda_{\text{iou}}, \lambda_{\text{L1}} \in \mathbb{R}$ are hyperparameters. These two losses are normalized by the number of objects inside the batch.

3.2 DETR architecture

The overall DETR architecture is surprisingly simple and depicted in Figure 2. It contains three main components, which we describe below: a CNN backbone to extract a compact feature representation, an encoder-decoder transformer, and a simple feed forward network (FFN) that makes the final detection prediction.

Unlike many modern detectors, DETR can be implemented in any deep learning framework that provides a common CNN backbone and a transformer architecture implementation with just a few hundred lines. Inference code for DETR can be implemented in less than 50 lines in PyTorch [32]. We hope that the simplicity of our method will attract new researchers to the detection community.

Backbone. Starting from the initial image $x_{\text{img}} \in \mathbb{R}^{3 \times H_0 \times W_0}$ (with 3 color channels²), a conventional CNN backbone generates a lower-resolution activation map $f \in \mathbb{R}^{C \times H \times W}$. Typical values we use are $C = 2048$ and $H, W = \frac{H_0}{32}, \frac{W_0}{32}$.

Transformer encoder. First, a 1x1 convolution reduces the channel dimension of the high-level activation map f from C to a smaller dimension d , creating a new feature map $z_0 \in \mathbb{R}^{d \times H \times W}$. The encoder expects a sequence as input, hence we collapse the spatial dimensions of z_0 into one dimension, resulting in a $d \times HW$ feature map. Each encoder layer has a standard architecture and consists of a multi-head self-attention module and a feed forward network (FFN). Since the transformer architecture is permutation-invariant, we supplement it with fixed positional encodings [31, 3] that are added to the input of each attention layer. We defer to the supplementary material the detailed definition of the architecture, which follows the one described in [47].

² The input images are batched together, applying 0-padding adequately to ensure they all have the same dimensions (H_0, W_0) as the largest image of the batch.

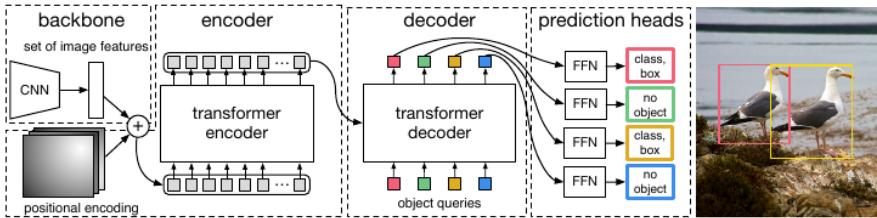


図2:DETRは従来のCNNバックボーンを用いて、入力画像の2次元表現を学習する。モデルは、変換エンコーダに渡す前に、それを平坦化し、位置エンコーディングで補足する。変換デコーダは、学習された少数の固定された位置埋め込みを入力とし、これをオブジェクトクエリと呼び、さらにエンコーダ出力に注目する。デコーダの各出力埋め込みを、検出(クラスとバウンディングボックス)または「オブジェクトなし」クラスのいずれかを予測する共有フィードフォワードネットワーク(FFN)に渡す。

トランسفォーマーデコーダ。デコーダはトランسفォーマーの標準的なアーキテクチャに従い、多頭の自己とエンコーダ・デコーダの注意メカニズムを用いて、サイズ d の N 個の埋め込みを変換する。オリジナルの変換器との違いは、我々のモデルが各デコーダ層で N 個のオブジェクトを並列にデコードするのに対し、Vaswaniら[47]は出力シーケンスを一度に1要素ずつ予測する自己回帰モデルを用いていることである。この概念に馴染みのない読者には、補足資料を参照されたい。デコーダも順列不变であるため、異なる結果を得るために N 個の入力埋め込みが異なる必要がある。これらの入力埋め込みは、オブジェクトクエリと呼ばれる学習された位置エンコーディングであり、エンコーダと同様に、各注意層の入力に追加する。 N 個のオブジェクトクエリはデコーダによって出力埋め込みに変換される。次に、フィードフォワードネットワーク(次のサブセクションで説明)によって、箱座標とクラスラベルに独立にデコードされ、 N 個の最終予測値が得られる。これらの埋め込みに対する自己とエンコーダ・デコーダの注意を用いることで、画像全体をコンテキストとして使用することができながら、モデルはすべてのオブジェクトと一緒にペアワイズ関係を使用して推論する。

予測フィードフォワードネットワーク(FFN)。最終的な予測は、ReLU活性化関数と隠れ次元 d を持つ3層パーセプトロンと、線形射影層によって計算される。FFNは入力画像に対して正規化された中心座標、箱の高さ、幅を予測し、線形層はソフトマックス関数を用いてクラスラベルを予測する。 N 個のバウンディングボックスの固定サイズの集合を予測するので、 N は通常、画像中の実際の注目オブジェクトの数よりも大きいので、スロット内でオブジェクトが検出されないことを表すために、追加の特殊クラスラベル \varnothing が使用される。このクラスは、標準的な物体検出アプローチにおける「背景」クラスと同様の役割を果たす。

補助的なデコーディングロス。学習時にデコーダに補助損失[1]を用いること、特にモデルが各クラスの正しい数のオブジェクトを出力するのに役立つことがわかった。

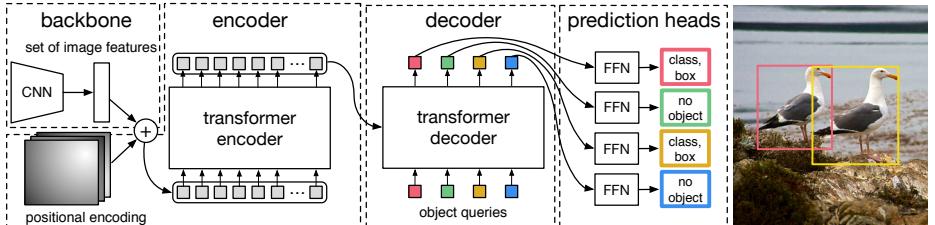


Fig. 2: DETR uses a conventional CNN backbone to learn a 2D representation of an input image. The model flattens it and supplements it with a positional encoding before passing it into a transformer encoder. A transformer decoder then takes as input a small fixed number of learned positional embeddings, which we call *object queries*, and additionally attends to the encoder output. We pass each output embedding of the decoder to a shared feed forward network (FFN) that predicts either a detection (class and bounding box) or a “*no object*” class.

Transformer decoder. The decoder follows the standard architecture of the transformer, transforming N embeddings of size d using multi-headed self- and encoder-decoder attention mechanisms. The difference with the original transformer is that our model decodes the N objects in parallel at each decoder layer, while Vaswani et al. [47] use an autoregressive model that predicts the output sequence one element at a time. We refer the reader unfamiliar with the concepts to the supplementary material. Since the decoder is also permutation-invariant, the N input embeddings must be different to produce different results. These input embeddings are learnt positional encodings that we refer to as *object queries*, and similarly to the encoder, we add them to the input of each attention layer. The N object queries are transformed into an output embedding by the decoder. They are then *independently* decoded into box coordinates and class labels by a feed forward network (described in the next subsection), resulting N final predictions. Using self- and encoder-decoder attention over these embeddings, the model globally reasons about all objects together using pair-wise relations between them, while being able to use the whole image as context.

Prediction feed-forward networks (FFNs). The final prediction is computed by a 3-layer perceptron with ReLU activation function and hidden dimension d , and a linear projection layer. The FFN predicts the normalized center coordinates, height and width of the box w.r.t. the input image, and the linear layer predicts the class label using a softmax function. Since we predict a fixed-size set of N bounding boxes, where N is usually much larger than the actual number of objects of interest in an image, an additional special class label \emptyset is used to represent that no object is detected within a slot. This class plays a similar role to the “background” class in the standard object detection approaches.

Auxiliary decoding losses. We found helpful to use auxiliary losses [1] in decoder during training, especially to help the model output the correct number

各デコーダ層の後に予測FFNとハンガリー損失を追加する。すべての予測FFNはパラメータを共有する。異なるデコーダ層からの予測FFNへの入力を正規化するために、追加の共有層ノルムを使用する。

4 Experiments

DETRはFaster R-CNNと比較して、COCOでの定量評価において競争力のある結果を達成することを示す。次に、アーキテクチャと損失に関する詳細なアプレーション研究を行い、洞察と定性的な結果を得る。最後に、DETRが汎用的で拡張可能なモデルであることを示すために、固定されたDETRモデルに対して小さな拡張のみを訓練した、パノプティックセグメンテーションの結果を示す。<https://github.com/facebookresearch/detr>で実験を再現するためのコードと事前学習済みモデルを提供する。

データセット COCO 2017検出データセットとパノプティックセグメンテーションデータセット[24, 18]で実験を行い、118k枚のトレーニング画像と5k枚の検証画像を含む。各画像はバウンディングボックスとパノプティックセグメンテーションでアノテーションされている。画像あたり平均7インスタンスあり、トレーニングセットの1つの画像には最大63インスタンスあり、同じ画像上で小さいものから大きいものまである。指定がない場合、APを複数の閾値に対する積分メトリックであるbbox APとして報告する。Faster R-CNNとの比較のために、最後のトレーニングエポックでの検証APを報告し、アプレーションについては、最後の10エポックからの検証結果に対する中央値を報告する。

技術的詳細 AdamW[26]を用いてDETRを学習し、初期変換器の学習率を 10^{-4} 、バックボーンの学習率を 10^{-5} 、重み減衰を 10^{-4} とする。すべての変換重みはXavier init [11]で初期化され、バックボーンはtorchvisionのImageNet-pretrained ResNet model [15]で凍結バッチノルム層で初期化される。ResNet50とResNet-101の2種類のバックボーンを用いた結果を報告する。対応するモデルをそれぞれDETR、DETR-R101と呼ぶ。また、[21]に従い、バックボーンの最終段に拡張を加え、この段の最初の畳み込みからストライドを削除することで、特徴解像度を向上させる。対応するモデルをそれぞれDETR-DC5、DETR-DC5-R101(拡張C5ステージ)と呼ぶ。この修正により、解像度が2倍向上し、小さなオブジェクトの性能が向上するが、エンコーダの自己アテンションのコストが16倍高くなり、全体として計算コストが2倍増加する。これらのモデルとFaster R-CNNのFLOPsの完全な比較を表1に示す。

スケール拡張を使用し、最短辺が少なくとも480ピクセル、最大800ピクセル、最長辺が最大1333ピクセルになるように入力画像をリサイズする[50]。エンコーダの自己注意によるグローバルな関係の学習を助けるために、学習中にランダムクロップ補強も適用し、性能を約1AP向上させる。具体的には、訓練画像は確率0.5でランダムな矩形パッチに切り取られ、その後800-1333にリサイズされる。変換器はデフォルトのドロップアウト0.1で学習される。推論時に、いくつかのスロットは空のクラスを予測する。

of objects of each class. We add prediction FFNs and Hungarian loss after each decoder layer. All predictions FFNs share their parameters. We use an additional shared layer-norm to normalize the input to the prediction FFNs from different decoder layers.

4 Experiments

We show that DETR achieves competitive results compared to Faster R-CNN in quantitative evaluation on COCO. Then, we provide a detailed ablation study of the architecture and loss, with insights and qualitative results. Finally, to show that DETR is a versatile and extensible model, we present results on panoptic segmentation, training only a small extension on a fixed DETR model. We provide code and pretrained models to reproduce our experiments at <https://github.com/facebookresearch/detr>.

Dataset. We perform experiments on COCO 2017 detection and panoptic segmentation datasets [24,18], containing 118k training images and 5k validation images. Each image is annotated with bounding boxes and panoptic segmentation. There are 7 instances per image on average, up to 63 instances in a single image in training set, ranging from small to large on the same images. If not specified, we report AP as bbox AP, the integral metric over multiple thresholds. For comparison with Faster R-CNN we report validation AP at the last training epoch, for ablations we report median over validation results from the last 10 epochs.

Technical details. We train DETR with AdamW [26] setting the initial transformer’s learning rate to 10^{-4} , the backbone’s to 10^{-5} , and weight decay to 10^{-4} . All transformer weights are initialized with Xavier init [11], and the backbone is with ImageNet-pretrained ResNet model [15] from TORCHVISION with frozen batchnorm layers. We report results with two different backbones: a ResNet-50 and a ResNet-101. The corresponding models are called respectively DETR and DETR-R101. Following [21], we also increase the feature resolution by adding a dilation to the last stage of the backbone and removing a stride from the first convolution of this stage. The corresponding models are called respectively DETR-DC5 and DETR-DC5-R101 (dilated C5 stage). This modification increases the resolution by a factor of two, thus improving performance for small objects, at the cost of a 16x higher cost in the self-attentions of the encoder, leading to an overall 2x increase in computational cost. A full comparison of FLOPs of these models and Faster R-CNN is given in Table 1.

We use scale augmentation, resizing the input images such that the shortest side is at least 480 and at most 800 pixels while the longest at most 1333 [50]. To help learning global relationships through the self-attention of the encoder, we also apply random crop augmentations during training, improving the performance by approximately 1 AP. Specifically, a train image is cropped with probability 0.5 to a random rectangular patch which is then resized again to 800-1333. The transformer is trained with default dropout of 0.1. At inference

表1:COCO検証セットにおけるResNet-50とResNet-101バックボーンを用いたFaster R-CNNとの比較。上段はDetectron2[50]のFaster R-CNNモデルの結果、中段はGIoU[38]のFaster R-CNNモデルの結果、ランダムなクロップの訓練時間増強、9倍の長い訓練スケジュールの結果である。DETRモデルは、 AP_S は低いが AP_L は大幅に改善され、大きく調整されたFaster R-CNNベースラインと同等の結果を達成した。FLOPSとFPSの測定には、torchscript Faster R-CNNとDETRモデルを使用する。名前にR101がない結果はResNet-50に対応する。

Model	GFLOPS/FPS	#params	AP	AP ₅₀	AP ₇₅	AP _S	AP _M	AP _L
Faster RCNN-DC5	320/16	166M	39.0	60.5	42.3	21.4	43.5	52.5
Faster RCNN-FPN	180/26	42M	40.2	61.0	43.8	24.2	43.5	52.0
Faster RCNN-R101-FPN	246/20	60M	42.0	62.5	45.9	25.2	45.6	54.6
Faster RCNN-DC5+	320/16	166M	41.1	61.4	44.3	22.9	45.9	55.0
Faster RCNN-FPN+	180/26	42M	42.0	62.1	45.5	26.6	45.4	53.4
Faster RCNN-R101-FPN+	246/20	60M	44.0	63.9	47.8	27.2	48.1	56.0
DETR	86/28	41M	42.0	62.4	44.2	20.5	45.8	61.1
DETR-DC5	187/12	41M	43.3	63.1	45.9	22.5	47.3	61.1
DETR-R101	152/20	60M	43.5	63.8	46.4	21.9	48.0	61.8
DETR-DC5-R101	253/10	60M	44.9	64.7	47.7	23.7	49.5	62.3

APを最適化するために、これらのスロットの予測値を、対応する信頼度を用いて、2番目にスコアの高いクラスで上書きする。空のスロットをフィルタリングする場合と比較して、APを2ポイント改善する。その他の学習ハイパーパラメータはセクションA.4にある。アプリケーション実験では、300エポックの学習スケジュールを使用し、200エポック後に学習率を10分の1に低下させる。16個のV100 GPUで300エポックのベースラインモデルのトレーニングに3日かかり、GPUあたり4枚の画像が必要である(そのため、総バッチサイズは64枚)。Faster R-CNNとの比較に使用した長いスケジュールでは、500エポック学習し、400エポック後に学習率を低下させた。このスケジュールは、短いスケジュールに比べて1.5AP追加される。

4.1 より高速なR-CNNとの比較

トランスフォーマーは通常、非常に長い学習スケジュールとドロップアウトを持つAdamまたはAdagradオプティマイザーで学習され、これはDETRにも当てはまる。しかし、より高速なR-CNNは、最小限のデータ増強でSGDを用いて学習されるため、Adamやdropoutの応用が成功していることは知られていない。このような違いがあるにもかかわらず、我々はFaster R-CNNベースラインをより強くすることを試みる。DETRと整合させるために、ボックスロスに一般化IoU [38]を追加し、同じランダムクロップ増強と、結果を改善することで知られる長いトレーニング[13]を行う。結果を表1に示す。上段では、3xスケジュールで学習したモデルに対するDetectron2 Model Zoo [50]のFaster R-CNNの結果を示す。

Table 1: Comparison with Faster R-CNN with a ResNet-50 and ResNet-101 backbones on the COCO validation set. The top section shows results for Faster R-CNN models in Detectron2 [50], the middle section shows results for Faster R-CNN models with GIoU [38], random crops train-time augmentation, and the long 9x training schedule. DETR models achieve comparable results to heavily tuned Faster R-CNN baselines, having lower AP_S but greatly improved AP_L. We use torchscript Faster R-CNN and DETR models to measure FLOPS and FPS. Results without R101 in the name correspond to ResNet-50.

Model	GFLOPS/FPS	#params	AP	AP ₅₀	AP ₇₅	AP _S	AP _M	AP _L
Faster RCNN-DC5	320/16	166M	39.0	60.5	42.3	21.4	43.5	52.5
Faster RCNN-FPN	180/26	42M	40.2	61.0	43.8	24.2	43.5	52.0
Faster RCNN-R101-FPN	246/20	60M	42.0	62.5	45.9	25.2	45.6	54.6
Faster RCNN-DC5+	320/16	166M	41.1	61.4	44.3	22.9	45.9	55.0
Faster RCNN-FPN+	180/26	42M	42.0	62.1	45.5	26.6	45.4	53.4
Faster RCNN-R101-FPN+	246/20	60M	44.0	63.9	47.8	27.2	48.1	56.0
DETR	86/28	41M	42.0	62.4	44.2	20.5	45.8	61.1
DETR-DC5	187/12	41M	43.3	63.1	45.9	22.5	47.3	61.1
DETR-R101	152/20	60M	43.5	63.8	46.4	21.9	48.0	61.8
DETR-DC5-R101	253/10	60M	44.9	64.7	47.7	23.7	49.5	62.3

time, some slots predict empty class. To optimize for AP, we override the prediction of these slots with the second highest scoring class, using the corresponding confidence. This improves AP by 2 points compared to filtering out empty slots. Other training hyperparameters can be found in section A.4. For our ablation experiments we use training schedule of 300 epochs with a learning rate drop by a factor of 10 after 200 epochs, where a single epoch is a pass over all training images once. Training the baseline model for 300 epochs on 16 V100 GPUs takes 3 days, with 4 images per GPU (hence a total batch size of 64). For the longer schedule used to compare with Faster R-CNN we train for 500 epochs with learning rate drop after 400 epochs. This schedule adds 1.5 AP compared to the shorter schedule.

4.1 Comparison with Faster R-CNN

Transformers are typically trained with Adam or Adagrad optimizers with very long training schedules and dropout, and this is true for DETR as well. Faster R-CNN, however, is trained with SGD with minimal data augmentation and we are not aware of successful applications of Adam or dropout. Despite these differences we attempt to make a Faster R-CNN baseline stronger. To align it with DETR, we add generalized IoU [38] to the box loss, the same random crop augmentation and long training known to improve results [13]. Results are presented in Table 1. In the top section we show Faster R-CNN results from Detectron2 Model Zoo [50] for models trained with the 3x schedule. In the middle section we show results (with a “+”) for the same models but trained

表2:エンコーダサイズの効果各行は、エンコーダ層の数を変化させ、デコーダ層の数を固定したモデルに対応する。エンコーダ層を増やすと、性能は徐々に向上する。

#layers	GFLOPS/FPS	#params	AP	AP ₅₀	AP _S	AP _M	AP _L
0	76/28	33.4M	36.7	57.4	16.8	39.6	54.2
3	81/25	37.4M	40.1	60.6	18.5	43.8	58.6
6	86/23	41.3M	40.6	61.6	19.9	44.3	60.2
12	95/20	49.2M	41.6	62.1	19.8	44.9	61.9

中段では、同じモデルを9倍のスケジュール(109エポック)で学習させ、合計で1-2APを追加した強化についての結果("+"を示す。表1の最後のセクションでは、複数のDETRモデルの結果を示す。パラメータ数で比較できるように、幅256の6つの変換層と6つのデコーダ層を持つモデルを8つの注意ヘッドで選択する。FPNを用いたFaster R-CNNと同様に、このモデルは41.3Mのパラメータを持ち、そのうち23.5MがResNet-50、17.8Mが変換器である。Faster R-CNNとDETRは、学習時間が長くなるにつれてさらに改善される可能性が高いが、DETRは同じパラメータ数でFaster R-CNNと競合することができ、COCO valサブセットで42APを達成すると結論づけることができる。DETRがこれを実現する方法はAP_Lを改善すること(+7.8)であるが、AP_S(-5.5)ではモデルがまだ遅れをとっていることに注意。同じパラメータ数で同程度のFLOP数を持つDETR-DC5はAPが高いが、AP_Sでも有意に遅れている。ResNet-101をバックボーンとする高速R-CNNとDETRも同等の結果を示す。

4.2 Ablations

変換デコーダにおける注意メカニズムは、異なる検出の特徴表現間の関係をモデル化する重要な構成要素である。アブレーション解析では、アーキテクチャの他の構成要素と損失が最終的な性能にどのような影響を与えるかを探る。本研究では、6エンコーダ、6デコーダ層、幅256のResNet-50ベースのDETRモデルを選択した。このモデルは41.3Mのパラメータを持ち、短いスケジュールで40.6、長いスケジュールで42.0APを達成し、同じバックボーンを持つFaster R-CNN-FPNと同様に28FPSで動作する。

エンコーダ層の数。エンコーダ層の数を変えることで、グローバルな画像レベルの自己アテンションの重要性を評価する(表2)。エンコーダ層がない場合、全体のAPは3.9ポイント低下し、大きなオブジェクトでは6.0APと、より顕著に低下する。我々は、グローバルなシーン推論を用いることで、エンコーダがオブジェクトの分離に重要であるという仮説を立てた。図3では、学習済みモデルの最後のエンコーダ層のアテンションマップを、画像中のいくつかの点に着目して可視化している。エンコーダは既にインスタンスを分離しているようで、デコーダのオブジェクトの抽出とローカライズが単純化されている可能性が高い。

デコーダ層の数。各デコーディング層の後に補助損失を適用するため(セクション3.2参照)、予測FFNはpreFig.3:参照点の集

Table 2: Effect of encoder size. Each row corresponds to a model with varied number of encoder layers and fixed number of decoder layers. Performance gradually improves with more encoder layers.

#layers	GFLOPS/FPS	#params	AP	AP ₅₀	AP _S	AP _M	AP _L
0	76/28	33.4M	36.7	57.4	16.8	39.6	54.2
3	81/25	37.4M	40.1	60.6	18.5	43.8	58.6
6	86/23	41.3M	40.6	61.6	19.9	44.3	60.2
12	95/20	49.2M	41.6	62.1	19.8	44.9	61.9

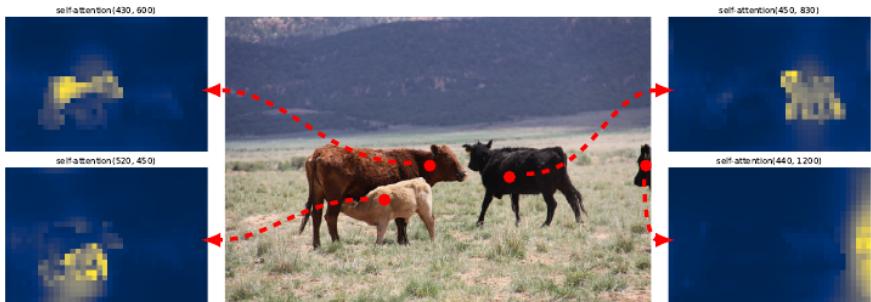
with the 9x schedule (109 epochs) and the described enhancements, which in total adds 1-2 AP. In the last section of Table 1 we show the results for multiple DETR models. To be comparable in the number of parameters we choose a model with 6 transformer and 6 decoder layers of width 256 with 8 attention heads. Like Faster R-CNN with FPN this model has 41.3M parameters, out of which 23.5M are in ResNet-50, and 17.8M are in the transformer. Even though both Faster R-CNN and DETR are still likely to further improve with longer training, we can conclude that DETR can be competitive with Faster R-CNN with the same number of parameters, achieving 42 AP on the COCO val subset. The way DETR achieves this is by improving AP_L (+7.8), however note that the model is still lagging behind in AP_S (-5.5). DETR-DC5 with the same number of parameters and similar FLOP count has higher AP, but is still significantly behind in AP_S too. Faster R-CNN and DETR with ResNet-101 backbone show comparable results as well.

4.2 Ablations

Attention mechanisms in the transformer decoder are the key components which model relations between feature representations of different detections. In our ablation analysis, we explore how other components of our architecture and loss influence the final performance. For the study we choose ResNet-50-based DETR model with 6 encoder, 6 decoder layers and width 256. The model has 41.3M parameters, achieves 40.6 and 42.0 AP on short and long schedules respectively, and runs at 28 FPS, similarly to Faster R-CNN-FPN with the same backbone.

Number of encoder layers. We evaluate the importance of global image-level self-attention by changing the number of encoder layers (Table 2). Without encoder layers, overall AP drops by 3.9 points, with a more significant drop of 6.0 AP on large objects. We hypothesize that, by using global scene reasoning, the encoder is important for disentangling objects. In Figure 3, we visualize the attention maps of the last encoder layer of a trained model, focusing on a few points in the image. The encoder seems to separate instances already, which likely simplifies object extraction and localization for the decoder.

Number of decoder layers. We apply auxiliary losses after each decoding layer (see Section 3.2), hence, the prediction FFNs are trained by design to pre-



合に対するエンコーダの自己アテンションを設計により学習する。エンコーダは個々のインスタンスを分離することができる。検証セット画像に対して、ベースラインDETRモデルで予測を行う。

各デコーダ層の出力からオブジェクトを決定する。デコーディングの各段階で予測されるオブジェクトを評価することで、各デコーダー層の重要性を分析する(図4)。APと AP_{50} はいずれも各層で改善し、第1層と最終層で合計+8.2/9.5APの非常に大きな改善となる。集合ベースの損失を持つDETRは、設計上NMSを必要としない。これを検証するために、各デコーダ後の出力に対してデフォルトのパラメータ[50]で標準的なNMS手順を実行する。NMSは最初のデコーダからの予測の性能向上させる。これは、変換器の単一の復号層では出力要素間の相互相関を計算できないため、同じオブジェクトに対して複数の予測を行いやすいという事実によって説明することができる。2層目以降の層では、活性化に対する自己注意のメカニズムにより、モデルは重複予測を抑制することができる。NMSによってもたらされる改善は、深さが増すにつれて減少することが観察される。最後の層では、NMSが真正予測を誤って除去するため、APの損失が小さいことが観察される。

エンコーダの注意を可視化するのと同様に、デコーダの注意も可視化したのが図6で、予測された各オブジェクトの注意マップを異なる色で着色している。デコーダの注意はかなり局所的であり、頭や脚などの物体の四肢にはほとんど注意を向けることを意味する。我々は、エンコーダがグローバルな注意によってインスタンスを分離した後、デコーダはクラスとオブジェクトの境界を抽出するために四肢にのみ注意すればよいという仮説を立てた。

FFNの重要性。FFN内部のトランスフォーマーは 1×1 畳み込み層と見ることができ、エンコーダーを注意増強畳み込みネットワーク[3]に類似させることができる。我々は、変換層に注意だけを残して、完全に削除することを試みる。ネットワークパラメータ数を41.3Mから28.7Mに減らし、トランスフォーマーに10.8Mしか残さないようにすると、性能は2.3AP低下し、FFNは良い結果を得るために重要であると結論づけられる。

位置エンコーディングの重要性我々のモデルには2種類の位置エンコーディングがある：空間位置エンコーディングと出力位置エンコーディング(オブジェクトクエリ)。

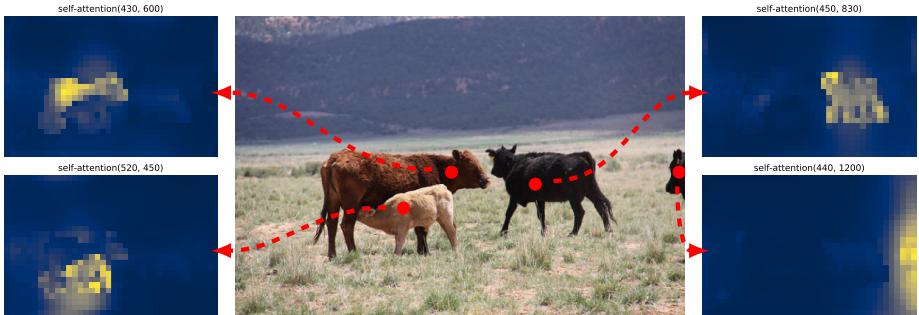


Fig. 3: Encoder self-attention for a set of reference points. The encoder is able to separate individual instances. Predictions are made with baseline DETR model on a validation set image.

dict objects out of the outputs of every decoder layer. We analyze the importance of each decoder layer by evaluating the objects that would be predicted at each stage of the decoding (Fig. 4). Both AP and AP_{50} improve after every layer, totalling into a very significant +8.2/9.5 AP improvement between the first and the last layer. With its set-based loss, DETR does not need NMS by design. To verify this we run a standard NMS procedure with default parameters [50] for the outputs after each decoder. NMS improves performance for the predictions from the first decoder. This can be explained by the fact that a single decoding layer of the transformer is not able to compute any cross-correlations between the output elements, and thus it is prone to making multiple predictions for the same object. In the second and subsequent layers, the self-attention mechanism over the activations allows the model to inhibit duplicate predictions. We observe that the improvement brought by NMS diminishes as depth increases. At the last layers, we observe a small loss in AP as NMS incorrectly removes true positive predictions.

Similarly to visualizing encoder attention, we visualize decoder attentions in Fig. 6, coloring attention maps for each predicted object in different colors. We observe that decoder attention is fairly local, meaning that it mostly attends to object extremities such as heads or legs. We hypothesize that after the encoder has separated instances via global attention, the decoder only needs to attend to the extremities to extract the class and object boundaries.

Importance of FFN. FFN inside transformers can be seen as 1×1 convolutional layers, making encoder similar to attention augmented convolutional networks [3]. We attempt to remove it completely leaving only attention in the transformer layers. By reducing the number of network parameters from 41.3M to 28.7M, leaving only 10.8M in the transformer, performance drops by 2.3 AP, we thus conclude that FFN are important for achieving good results.

Importance of positional encodings. There are two kinds of positional encodings in our model: spatial positional encodings and output positional encod-

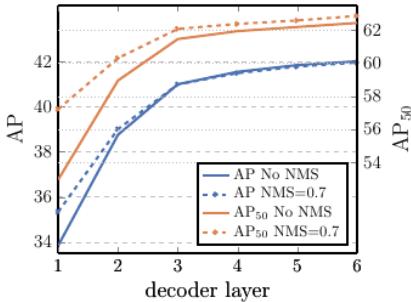


図4: 各デコーダ層後のAPとAP₅₀の性能。単一のロングスケジュールベースラインモデルが評価される。DETRは設計上NMSを必要としないが、これはこの図によって検証された。NMSは最終層のAPを下げ、TP予測を削除するが、最初のデコーダ層のAPを改善し、最初の層には通信がないため、二重予測を削除し、AP₅₀をわずかに改善する。



図5: 希少クラスに対する分布外汎化。学習セットに13個以上のキリンを持つ画像がないにもかかわらず、DETRは同じクラスの24個以上のインスタンスに汎化することに困難はない。

固定エンコーディングと学習エンコーディングの様々な組み合わせで実験した結果を表3に示す。出力位置エンコーディングは必要であり、削除することはできないので、デコーダ入力で1回渡すか、デコーダの注意層ごとにクエリに追加するかを実験する。最初の実験では、空間位置エンコーディングを完全に除去し、入力で出力位置エンコーディングを渡す。興味深いことに、このモデルは依然として32AP以上を達成し、ベースラインに対して7.8APを失う。次に、元の変換器[47]と同様に、固定正弦空間位置エンコーディングと出力エンコーディングを入力時に1回通過させ、位置エンコーディングを注意で直接通過させた場合と比較して、1.4AP低下することを発見した。注意に渡される学習された空間エンコーディングは、同様の結果を与える。意外なことに、エンコーダで空間エンコーディングを通過させないと、APが1.3APとわずかに低下するだけであることがわかった。エンコーディングをアテンションに渡すと、全てのレイヤーで共有され、出力エンコーディング(オブジェクトクエリ)は常に学習される。

これらのアプリケーションを考慮すると、トランスフォーマーの構成要素である、エンコーダーのグローバルな自己注意、FFN、複数のデコーダー層、位置エンコーディングはすべて、最終的な物体検出性能に大きく寄与していると結論づけられる。

損失の除去。マッチングコストと損失の異なる構成要素の重要性を評価するために、いくつかのモデルを訓練し、それらをオンとオフにする。分類損失、 χ_1 バウンディングボックス距離損失、GIoU [38]損失である。分類損失は学習に不可欠であり、オフにすることはできないため、バウンディングボックス距離損失なしのモデルとGIoU損失なしのモデルを学習し、3つの損失すべてで学習したベースラインと比較する。結果は表4に示されている。GIoUの損失は自社の勘定科目

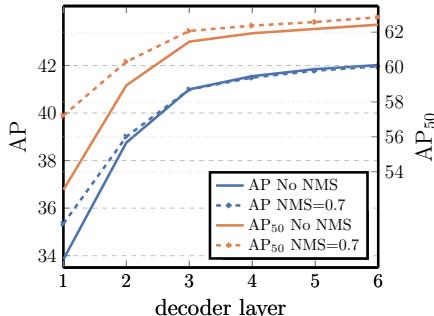


Fig. 4: AP and AP₅₀ performance after each decoder layer. A single long schedule baseline model is evaluated. DETR does not need NMS by design, which is validated by this figure. NMS lowers AP in the final layers, removing TP predictions, but improves AP in the first decoder layers, removing double predictions, as there is no communication in the first layer, and slightly improves AP₅₀.



Fig. 5: Out of distribution generalization for rare classes. Even though no image in the training set has more than 13 giraffes, DETR has no difficulty generalizing to 24 and more instances of the same class.

ings (object queries). We experiment with various combinations of fixed and learned encodings, results can be found in table 3. Output positional encodings are required and cannot be removed, so we experiment with either passing them once at decoder input or adding to queries at every decoder attention layer. In the first experiment we completely remove spatial positional encodings and pass output positional encodings at input and, interestingly, the model still achieves more than 32 AP, losing 7.8 AP to the baseline. Then, we pass fixed sine spatial positional encodings and the output encodings at input once, as in the original transformer [47], and find that this leads to 1.4 AP drop compared to passing the positional encodings directly in attention. Learned spatial encodings passed to the attentions give similar results. Surprisingly, we find that not passing any spatial encodings in the encoder only leads to a minor AP drop of 1.3 AP. When we pass the encodings to the attentions, they are shared across all layers, and the output encodings (object queries) are always learned.

Given these ablations, we conclude that transformer components: the global self-attention in encoder, FFN, multiple decoder layers, and positional encodings, all significantly contribute to the final object detection performance.

Loss ablations. To evaluate the importance of different components of the matching cost and the loss, we train several models turning them on and off. There are three components to the loss: classification loss, ℓ_1 bounding box distance loss, and GIoU [38] loss. The classification loss is essential for training and cannot be turned off, so we train a model without bounding box distance loss, and a model without the GIoU loss, and compare with baseline, trained with all three losses. Results are presented in table 4. GIoU loss on its own accounts

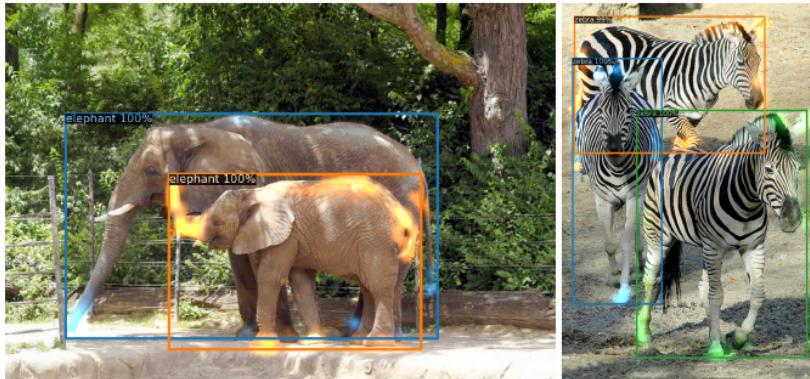


図6:予測されたオブジェクト(COCO val setの画像)に対するデコーダの注意の可視化。DETR-DC5モデルで予測。注意のスコアは、異なるオブジェクトに対して異なる色でコード化されている。デコーダは通常、脚や頭などの物体の四肢に注目する。カラーで見るのがベスト

表3: 異なる位置エンコーディングの結果を、正弦posを固定したベースライン(最後の行)と比較。エンコーダとデコーダの両方で、すべての注意層で渡されるエンコーディング。学習された埋め込みは全層で共有される。空間位置エンコーディングを使用しないと、APが大幅に低下する。興味深いことに、デコーダーに渡すと、APがわずかに低下するだけである。これらのモデルはすべて学習された出力位置エンコーディングを使用している。

spatial pos. enc. encoder	pos. enc. decoder	出力pos. enc. デコー ダ	AP	Δ	AP ₅₀	Δ
none	none	attnで学習した	32.8	-7.8	55.2	-6.5
sine at input	sine at input	入力で学習した。	39.2	-1.4	60.0	-1.6
learned at attn.	learned at attn.	attnで学習した。	39.6	-1.0	60.7	-0.9
none	sine at attn.	attnで学習した。	39.3	-1.3	60.3	-1.4
sine at attn.	sine at attn.	attnで学習した。	40.6	-	61.6	-

表4:APに対する損失成分の影響。 ℓ_1 損失とGIoU損失をオフにした2つのモデルを訓練し、 ℓ_1 単独では悪い結果を与えるが、GIoUと組み合わせるとAP_MとAP_Lが改善することを観察する。我々のベースライン(最後の行)は両方の損失を結合する。

class	ℓ_1	GIoU	AP	Δ	AP ₅₀	Δ	AP _S	AP _M	AP _L
✓	✓		35.8	-4.8	57.3	-4.4	13.7	39.8	57.9
✓		✓	39.9	-0.7	61.6	0	19.9	43.2	57.9
✓	✓	✓	40.6	-	61.6	-	19.9	44.3	60.2

は、ほとんどのモデル性能において、損失を組み合わせたベースラインに対して0.7APしか失わない。GIoUなしで ℓ_1 を使用すると、悪い結果になる。我々は

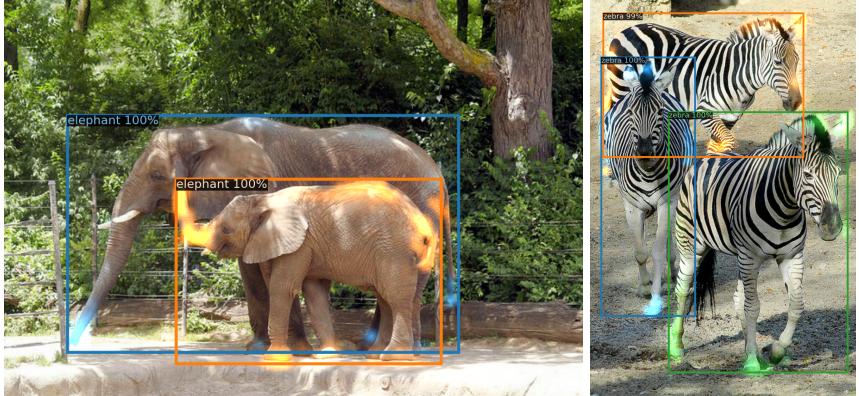


Fig. 6: Visualizing decoder attention for every predicted object (images from COCO `val` set). Predictions are made with DETR-DC5 model. Attention scores are coded with different colors for different objects. Decoder typically attends to object extremities, such as legs and heads. Best viewed in color.

Table 3: Results for different positional encodings compared to the baseline (last row), which has fixed sine pos. encodings passed at every attention layer in both the encoder and the decoder. Learned embeddings are shared between all layers. Not using spatial positional encodings leads to a significant drop in AP. Interestingly, passing them in decoder only leads to a minor AP drop. All these models use learned output positional encodings.

spatial pos. enc. encoder		output pos. enc. decoder		AP	Δ	AP ₅₀	Δ
	decoder		decoder				
none	none	learned at input		32.8	-7.8	55.2	-6.5
sine at input	sine at input	learned at input		39.2	-1.4	60.0	-1.6
learned at attn.	learned at attn.	learned at attn.		39.6	-1.0	60.7	-0.9
none	sine at attn.	learned at attn.		39.3	-1.3	60.3	-1.4
sine at attn.	sine at attn.	learned at attn.		40.6	-	61.6	-

Table 4: Effect of loss components on AP. We train two models turning off ℓ_1 loss, and GIoU loss, and observe that ℓ_1 gives poor results on its own, but when combined with GIoU improves AP_M and AP_L. Our baseline (last row) combines both losses.

class	ℓ_1	GIoU	AP	Δ	AP ₅₀	Δ	AP _S	AP _M	AP _L
✓	✓		35.8	-4.8	57.3	-4.4	13.7	39.8	57.9
✓		✓	39.9	-0.7	61.6	0	19.9	43.2	57.9
✓	✓	✓	40.6	-	61.6	-	19.9	44.3	60.2

for most of the model performance, losing only 0.7 AP to the baseline with combined losses. Using ℓ_1 without GIoU shows poor results. We only studied

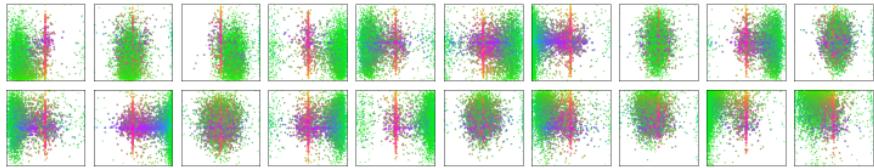


図7: DETRデコーダの全 $N = 100$ 予測スロットのうち 20 スロットについて、COCO 2017 val セットの全画像に対する全ボックス予測の可視化。各ボックス予測は、各画像サイズで正規化された 1×1 正方形の中心の座標を持つ点として表現される。点は、緑色が小さなボックス、赤色が大きな水平ボックス、青色が大きな垂直ボックスに対応するように色分けされている。各スロットは、いくつかの動作モードを持つ特定の領域とボックスサイズに特化することを学習することが観察される。COCOデータセットで一般的な、画像全体の大きなボックスを予測するモードが、ほぼ全てのスロットにあることに注意。

異なる損失を単純に除去する(毎回同じ重み付けをする)が、それらを組み合わせる他の手段では異なる結果が得られるかもしれない。

4.3 Analysis

デコーダ出力スロット解析 COCO 2017 val set の全画像について、異なるスロットによって予測されたボックスを図7に可視化する。DETRは各クエリスロットに対して異なる特殊化を学習する。各スロットには、異なるエリアとボックスサイズに焦点を当てた複数の動作モードがあることがわかる。特に、すべてのスロットは、画像全体のボックスを予測するためのモードを持っている(プロットの中央に赤い点が並んでいるのが見える)。これはCOCOにおけるオブジェクトの分布に関係しているという仮説を立てた。

未知のインスタンス数への汎化。COCOのいくつかのクラスは、同じ画像内の同じクラスの多くのインスタンスでうまく表現されていない。例えば、13匹以上のキリンがいる画像は学習セットに存在しない。DETRの汎化能力を検証するために、合成画像³を作成する(図5参照)。我々のモデルは、明らかに分布から外れた画像上の24頭のキリンをすべて見つけることができる。この実験により、各オブジェクトクエリに強いクラス特化がないことが確認された。

4.4 パノプティックセグメンテーションのためのDETR

パノプティックセグメンテーション[19]は、最近、コンピュータビジョンのコミュニティから多くの注目を集めている。Faster R-CNN [37]のMask R-CNN [14]への拡張と同様に、DETRはデコーダ出力の上にマスクヘッドを追加することで自然に拡張できる。このセクションでは、このような頭部を用いて、物と物のクラスを扱うことで、パノプティックセグメンテーション[19]を生成できることを示す。

³ Base picture credit: <https://www.piqsels.com/en/public-domain-photo-jzlwu>

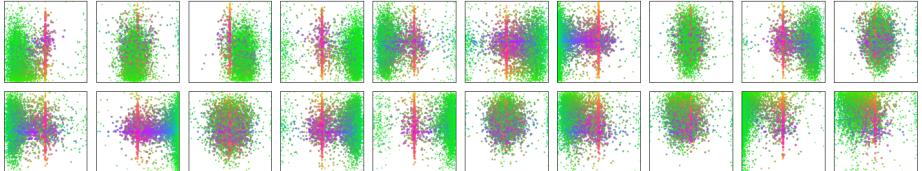


Fig. 7: Visualization of all box predictions on all images from COCO 2017 val set for 20 out of total $N = 100$ prediction slots in DETR decoder. Each box prediction is represented as a point with the coordinates of its center in the 1-by-1 square normalized by each image size. The points are color-coded so that green color corresponds to small boxes, red to large horizontal boxes and blue to large vertical boxes. We observe that each slot learns to specialize on certain areas and box sizes with several operating modes. We note that almost all slots have a mode of predicting large image-wide boxes that are common in COCO dataset.

simple ablations of different losses (using the same weighting every time), but other means of combining them may achieve different results.

4.3 Analysis

Decoder output slot analysis In Fig. 7 we visualize the boxes predicted by different slots for all images in COCO 2017 val set. DETR learns different specialization for each query slot. We observe that each slot has several modes of operation focusing on different areas and box sizes. In particular, all slots have the mode for predicting image-wide boxes (visible as the red dots aligned in the middle of the plot). We hypothesize that this is related to the distribution of objects in COCO.

Generalization to unseen numbers of instances. Some classes in COCO are not well represented with many instances of the same class in the same image. For example, there is no image with more than 13 giraffes in the training set. We create a synthetic image³ to verify the generalization ability of DETR (see Figure 5). Our model is able to find all 24 giraffes on the image which is clearly out of distribution. This experiment confirms that there is no strong class-specialization in each object query.

4.4 DETR for panoptic segmentation

Panoptic segmentation [19] has recently attracted a lot of attention from the computer vision community. Similarly to the extension of Faster R-CNN [37] to Mask R-CNN [14], DETR can be naturally extended by adding a mask head on top of the decoder outputs. In this section we demonstrate that such a head can be used to produce panoptic segmentation [19] by treating stuff and thing classes

³ Base picture credit: <https://www.piqsels.com/en/public-domain-photo-jzlwu>

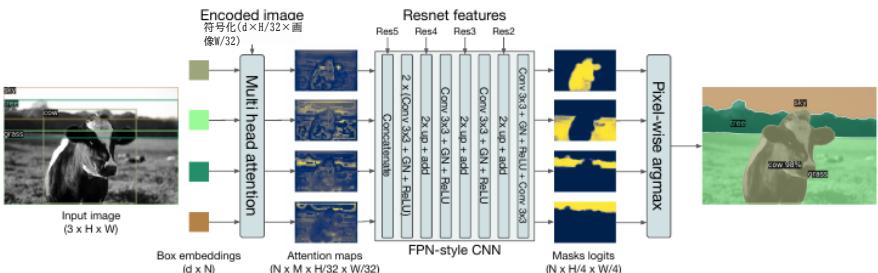


図8:パノプティックヘッドの説明図。検出された各オブジェクトに対して2値マスクが並列に生成され、ピクセル単位のargmaxを用いてマスクがマージされる。



図9:DETR-R101によって生成されたパノプティックセグメンテーションの定性的結果。DETRはモノとコトに対して統一的な方法で整列マスク予測を生成する。

を統一的に行う。COCOデータセットのパノプティックアノテーションに対して、80のモノのカテゴリに加え、53のモノのカテゴリを持つ実験を行った。

DETRを訓練し、COCO上のstuffとthingsの両方のクラスの周りのボックスを予測する。ハンガリー語のマッチングはボックス間の距離を用いて計算されるため、学習が可能であるためにはボックスの予測が必要である。また、予測されたボックスのそれについてバイナリマスクを予測するマスクヘッドを追加する(図8参照)。これは各オブジェクトの変換デコーダの出力を入力とし、エンコーダの出力に対するこの埋め込みのマルチヘッド(M個のヘッドを持つ)注目スコアを計算し、小さな解像度でオブジェクトごとにM個の注目ヒートマップを生成する。最終的な予測を行い、解像度を上げるために、FPNのようなアーキテクチャが使われる。アーキテクチャの詳細については、付録で説明する。マスクの最終解像度はストライド4で、各マスクはDICE/F-1損失[28]とFocal損失[23]を用いて独立に教師される。

マスクヘッドは、共同で学習するか、2段階のプロセスで学習することができる。ここでは、ボックスに対してのみDETRを学習し、その後、すべての重みを凍結し、マスクヘッドのみを25エポック学習させる。実験的には、これら2つのアプローチは同様の結果を与えるが、後者の方針を用いると、壁時計の総時間トレーニングが短くなるため、結果を報告する。

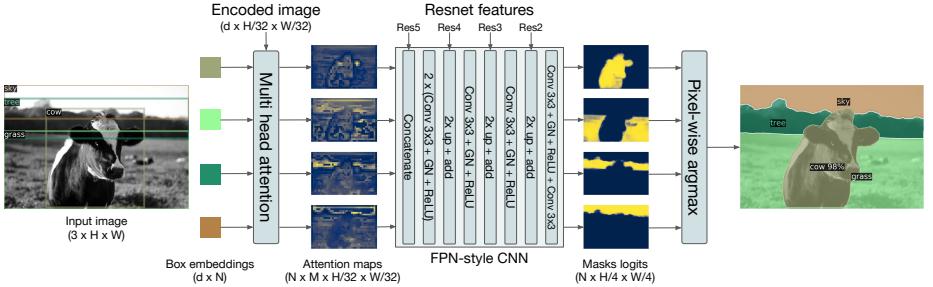


Fig. 8: Illustration of the panoptic head. A binary mask is generated in parallel for each detected object, then the masks are merged using pixel-wise argmax.

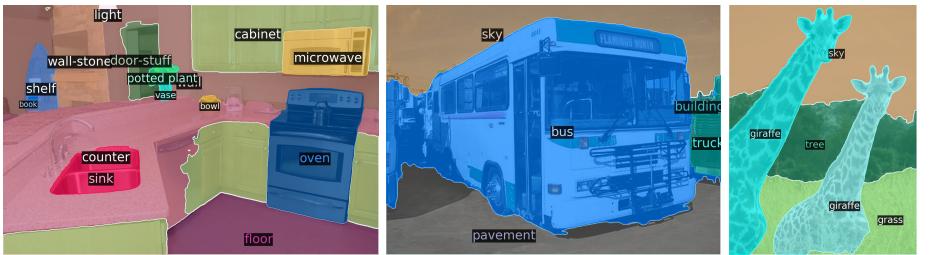


Fig. 9: Qualitative results for panoptic segmentation generated by DETR-R101. DETR produces aligned mask predictions in a unified manner for things and stuff.

in a unified way. We perform our experiments on the panoptic annotations of the COCO dataset that has 53 stuff categories in addition to 80 things categories.

We train DETR to predict boxes around both *stuff* and *things* classes on COCO, using the same recipe. Predicting boxes is required for the training to be possible, since the Hungarian matching is computed using distances between boxes. We also add a mask head which predicts a binary mask for each of the predicted boxes, see Figure 8. It takes as input the output of transformer decoder for each object and computes multi-head (with M heads) attention scores of this embedding over the output of the encoder, generating M attention heatmaps per object in a small resolution. To make the final prediction and increase the resolution, an FPN-like architecture is used. We describe the architecture in more details in the supplement. The final resolution of the masks has stride 4 and each mask is supervised independently using the DICE/F-1 loss [28] and Focal loss [23].

The mask head can be trained either jointly, or in a two steps process, where we train DETR for boxes only, then freeze all the weights and train only the mask head for 25 epochs. Experimentally, these two approaches give similar results, we report results using the latter method since it results in a shorter total wall-clock time training.

表5:COCO valデータセットにおける最先端手法UPSNNet [51]とPanoptic FPN [18]との比較 PanopticFPNをDETRと同じデータ補強で再トレーニングし、公平に比較できるように18倍のスケジュールで再トレーニングした。UPSNNetはlxスケジュールを使用し、UPSNNet-Mはマルチスケールテスト時間拡張を行ったバージョンである。

Model	Backbone	PQ	SQ	RQ	PQ th	SQ th	RQ th	PQ st	SQ st	RQ st	AP
PanopticFPN++	R50	42.4	79.3	51.6	49.2	82.4	58.8	32.3	74.8	40.6	37.7
UPSNNet	R50	42.5	78.0	52.5	48.6	79.4	59.6	33.4	75.9	41.7	34.3
UPSNNet-M	R50	43.0	79.1	52.8	48.9	79.7	59.7	34.1	78.2	42.3	34.3
PanopticFPN++	R101	44.1	79.5	53.3	51.0	83.2	60.6	33.6	74.0	42.1	39.7
DETR	R50	43.4	79.3	53.8	48.2	79.8	59.5	36.3	78.5	45.3	31.1
DETR-DC5	R50	44.6	79.8	55.0	49.4	80.5	60.6	37.3	78.7	46.5	31.9
DETR-R101	R101	45.1	79.9	55.5	50.5	80.9	61.7	37.0	78.5	46.0	33.0

最終的なパノプティックセグメンテーションを予測するために、各ピクセルのマスクスコアに対して単純にargmaxを使用し、対応するカテゴリを結果のマスクに割り当てる。この手順により、最終的なマスクは重なりを持たないことが保証されるため、DETRは異なるマスクの整列によく使われるヒューリスティック[19]を必要としない。

トレーニングの詳細 COCOデータセットのstuffとthingsクラスの周りのボックスを予測するために、バウンディングボックス検出のレシピに従ってDETR、DETR-DC5、DETR-R101モデルを訓練する。新しいマスクヘッドは25エポック学習される(詳細は補足を参照)。推論中、まず信頼度が85%以下の検出をフィルタリングし、次に画素毎のargmaxを計算し、各画素がどのマスクに属するかを決定する。次に、同じ物カテゴリの異なるマスク予測を1つにまとめ、空のもの(4ピクセル以下)をフィルタリングする。

主な結果。定性的な結果を図9に示す。表5では、我々の統一的なパノプティックセグメンテーションアプローチと、モノやコトを異なる形で扱ういくつかの確立された手法を比較している。パノプティック品質(PQ)と、物(PQth)と物(PQst)の分解を報告する。また、パノプティック後処理前(我々の場合はピクセル単位のargmaxを取る前)のマスクAP(モノのクラスで計算)も報告する。DETRがCOCO-val 2017で発表された結果や、我々の強力なPanopticFPNベースライン(公平な比較のため、DETRと同じデータ補強で訓練)を上回ることを示す。この結果の内訳は、DETRが特に物クラスで優位であることを示し、エンコーダの注意によって許される大域的な推論がこの結果の重要な要素であるという仮説を立てた。モノクラスでは、マスクAP計算のベースラインと比較して最大8mAPの深刻なデフィットにもかかわらず、DETRは競争力のあるPQthを得る。また、COCOデータセットのテストセットで本手法を評価したところ、46個のPQが得られた。我々のアプローチが、パノプティックセグメンテーションのための完全統一モデルの探求を刺激することを期待する。

Table 5: Comparison with the state-of-the-art methods UPSNet [51] and Panoptic FPN [18] on the COCO `val` dataset. We retrained PanopticFPN with the same data-augmentation as DETR, on a 18x schedule for fair comparison. UPSNet uses the `1x` schedule, UPSNet-M is the version with multiscale test-time augmentations.

Model	Backbone	PQ	SQ	RQ	PQ^{th}	SQ^{th}	RQ^{th}	PQ^{st}	SQ^{st}	RQ^{st}	AP
PanopticFPN++	R50	42.4	79.3	51.6	49.2	82.4	58.8	32.3	74.8	40.6	37.7
UPSnet	R50	42.5	78.0	52.5	48.6	79.4	59.6	33.4	75.9	41.7	34.3
UPSnet-M	R50	43.0	79.1	52.8	48.9	79.7	59.7	34.1	78.2	42.3	34.3
PanopticFPN++	R101	44.1	79.5	53.3	51.0	83.2	60.6	33.6	74.0	42.1	39.7
DETR	R50	43.4	79.3	53.8	48.2	79.8	59.5	36.3	78.5	45.3	31.1
DETR-DC5	R50	44.6	79.8	55.0	49.4	80.5	60.6	37.3	78.7	46.5	31.9
DETR-R101	R101	45.1	79.9	55.5	50.5	80.9	61.7	37.0	78.5	46.0	33.0

To predict the final panoptic segmentation we simply use an argmax over the mask scores at each pixel, and assign the corresponding categories to the resulting masks. This procedure guarantees that the final masks have no overlaps and, therefore, DETR does not require a heuristic [19] that is often used to align different masks.

Training details. We train DETR, DETR-DC5 and DETR-R101 models following the recipe for bounding box detection to predict boxes around stuff and things classes in COCO dataset. The new mask head is trained for 25 epochs (see supplementary for details). During inference we first filter out the detection with a confidence below 85%, then compute the per-pixel argmax to determine in which mask each pixel belongs. We then collapse different mask predictions of the same stuff category in one, and filter the empty ones (less than 4 pixels).

Main results. Qualitative results are shown in Figure 9. In table 5 we compare our unified panoptic segmenation approach with several established methods that treat things and stuff differently. We report the Panoptic Quality (PQ) and the break-down on things (PQ^{th}) and stuff (PQ^{st}). We also report the mask AP (computed on the things classes), before any panoptic post-treatment (in our case, before taking the pixel-wise argmax). We show that DETR outperforms published results on COCO-val 2017, as well as our strong PanopticFPN baseline (trained with same data-augmentation as DETR, for fair comparison). The result break-down shows that DETR is especially dominant on stuff classes, and we hypothesize that the global reasoning allowed by the encoder attention is the key element to this result. For things class, despite a severe deficit of up to 8 mAP compared to the baselines on the mask AP computation, DETR obtains competitive PQ^{th} . We also evaluated our method on the test set of the COCO dataset, and obtained 46 PQ. We hope that our approach will inspire the exploration of fully unified models for panoptic segmentation in future work.

5 Conclusion

DETRは、変換器と二分割マッチング損失に基づく物体検出システムのための新しい設計であり、直接集合予測のために提案された。このアプローチは、難易度の高いCOCOデータセットにおいて、最適化されたFaster R-CNNベースラインと同等の結果を達成した。DETRは実装が簡単で、パノプティックセグメンテーションに容易に拡張できる柔軟なアーキテクチャを持ち、競争力のある結果を得ることができる。また、Faster R-CNNよりも大きな物体に対して、自己注意によって実行されるグローバルな情報の処理のおかげで、有意に優れた性能を達成する。

この検出器の新しい設計には、特に小さな物体での学習、最適化、性能に関する新しい課題も伴う。現在の検出器は、同様の問題に対処するために数年の改良が必要であり、今後の研究でDETRへの対応が成功することが期待される。

6 謝辞

Sainbayar Sukhbaatar、Piotr Bojanowski、Natalia Neverova、Lopez-Paz、Guillaume Lample、Danielle Rothermel、Kaiming He、Ross Girshick、Xinlei Chen、そしてFacebook AI Research Parisチーム全体に感謝する。

References

1. Al-Rfou, R.、Choe, D.、Constant, N.、Guo, M.、Jones, L.: より深い自己注意を持つ文字レベルの言語モデリング。In: 人工知能に関するAAAI会議(2019)
2. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. In: ICLR (2015)
3. Bello, I., Zoph, B., Vaswani, A., Shlens, J., Le, Q.V.: Attention augmented convolutional networks. In: ICCV (2019)
4. Bodla, N., Singh, B., Chellappa, R., Davis, L.S.: Soft-NMS improving object detection with one line of code. In: ICCV (2017)
5. Cai, Z., Vasconcelos, N.: Cascade R-CNN: High quality object detection and instance segmentation. PAMI (2019)
6. Chan, W.、Saharia, C.、Hinton, G.、Norouzi, M.、Jaitly, N.: Impute r:arXiv:2002.08926(2020)Cordonnier, J.B.、Loukas, A.、Jaggi, M. :自己注意と畳み込み層の関係について。ICLR (2020)
7. BERT: 言語理解のための深い双方向変換器の事前学習。In: NAACL-HLT (2019)
8. Erhan, D., Szegedy, C., Toshev, A., Anguelov, D.: ディープニューラルネットワークを用いたスケーラブルな物体検出。で CVPR (2014)
10. Ghazvininejad, M., Levy, O., Liu, Y., Zettlemoyer, L.: Mask-predict: Parallel decoding of conditional masked language models. arXiv:1904.09324 (2019)
11. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: AISTATS (2010)

5 Conclusion

We presented DETR, a new design for object detection systems based on transformers and bipartite matching loss for direct set prediction. The approach achieves comparable results to an optimized Faster R-CNN baseline on the challenging COCO dataset. DETR is straightforward to implement and has a flexible architecture that is easily extensible to panoptic segmentation, with competitive results. In addition, it achieves significantly better performance on large objects than Faster R-CNN, likely thanks to the processing of global information performed by the self-attention.

This new design for detectors also comes with new challenges, in particular regarding training, optimization and performances on small objects. Current detectors required several years of improvements to cope with similar issues, and we expect future work to successfully address them for DETR.

6 Acknowledgements

We thank Sainbayar Sukhbaatar, Piotr Bojanowski, Natalia Neverova, David Lopez-Paz, Guillaume Lample, Danielle Rothermel, Kaiming He, Ross Girshick, Xinlei Chen and the whole Facebook AI Research Paris team for discussions and advices without which this work would not be possible.

References

1. Al-Rfou, R., Choe, D., Constant, N., Guo, M., Jones, L.: Character-level language modeling with deeper self-attention. In: AAAI Conference on Artificial Intelligence (2019)
2. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. In: ICLR (2015)
3. Bello, I., Zoph, B., Vaswani, A., Shlens, J., Le, Q.V.: Attention augmented convolutional networks. In: ICCV (2019)
4. Bodla, N., Singh, B., Chellappa, R., Davis, L.S.: Soft-NMS improving object detection with one line of code. In: ICCV (2017)
5. Cai, Z., Vasconcelos, N.: Cascade R-CNN: High quality object detection and instance segmentation. PAMI (2019)
6. Chan, W., Saharia, C., Hinton, G., Norouzi, M., Jaitly, N.: Imputer: Sequence modelling via imputation and dynamic programming. arXiv:2002.08926 (2020)
7. Cordonnier, J.B., Loukas, A., Jaggi, M.: On the relationship between self-attention and convolutional layers. In: ICLR (2020)
8. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: NAACL-HLT (2019)
9. Erhan, D., Szegedy, C., Toshev, A., Anguelov, D.: Scalable object detection using deep neural networks. In: CVPR (2014)
10. Ghazvininejad, M., Levy, O., Liu, Y., Zettlemoyer, L.: Mask-predict: Parallel decoding of conditional masked language models. arXiv:1904.09324 (2019)
11. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: AISTATS (2010)

12. Gu, J., Bradbury, J., Xiong, C., Li, V.O., Socher, R.: Non-autoregressive neural machine translation. In: ICLR (2018)
13. He, K., Girshick, R., Dollár, P.: Rethinking imagenet pre-training. In: ICCV (2019)
14. He, K., Gkioxari, G., Dollár, P., Girshick, R.B.: Mask R-CNN. In: ICCV (2017)
15. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)
16. Hosang, J.H., Benenson, R., Schiele, B.: Learning non-maximum suppression. In: CVPR (2017)
17. Hu, H., Gu, J., Zhang, Z., Dai, J., Wei, Y.: Relation networks for object detection. In: CVPR (2018)
18. Kirillov, A., Girshick, R., He, K., Dollár, P.: Panoptic feature pyramid networks. In: CVPR (2019)
19. Kirillov, A., He, K., Girshick, R., Rother, C., Dollar, P.: Panoptic segmentation. In: CVPR (2019)
20. Kuhn, H.W.: The hungarian method for the assignment problem (1955)
21. Li, Y., Qi, H., Dai, J., Ji, X., Wei, Y.: Fully convolutional instance-aware semantic segmentation. In: CVPR (2017)
22. Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: CVPR (2017)
23. Lin, T.Y., Goyal, P., Girshick, R.B., He, K., Dollár, P.: Focal loss for dense object detection. In: ICCV (2017)
24. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft COCO: Common objects in context. In: ECCV (2014)
25. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S.E., Fu, C.Y., Berg, A.C.: Ssd: Single shot multibox detector. In: ECCV (2016)
26. Loshchilov, I., Hutter, F.: 非結合重み減衰正則化. In: ICLR (2017) Lüsch
27. er, C., Beck, E., Irie, K., Kitza, M., Michel, W., Zeyer, A., Schlüter, R., Ney, H.: Rwth asr systems for librispeech:arXiv:1905.03072 (2019)
28. Milletari, F., Navab, N., Ahmadi, S.A.: V-net: Fully convolutional neural networks for volumetric medical image segmentation. In: 3DV (2016)
29. Oord, A.v.d., Li, Y., Babuschkin, I., Simonyan, K., Vinyals, O., Kavukcuoglu, K., Driessche, G.v.d., Lockhart, E., Cobo, L.C., Stimberg, F., et al.: Parallel wavenet: Fast high-fidelity speech synthesis. arXiv:1711.10433 (2017)
30. Park, E., Berg, A.C.: Learning to decompose for object detection and instance segmentation. arXiv:1511.06449 (2015)
31. Parmar, N., Vaswani, A., Uszkoreit, J., Kaiser, L., Shazeer, N., Ku, A., Tran, D.: Image transformer. In: ICML (2018)
32. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: Pytorch: An imperative style, high-performance deep learning library. In: NeurIPS (2019)
33. Pineda, L., Salvador, A., Drozdzal, M., Romero, A.: Elucidating image-to-set prediction: An analysis of models, losses and datasets. arXiv:1904.05709 (2019)
34. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I.: Language models are unsupervised multitask learners (2019)
35. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: CVPR (2016)
36. Ren, M., Zemel, R.S.: End-to-end instance segmentation with recurrent attention. In: CVPR (2017)

12. Gu, J., Bradbury, J., Xiong, C., Li, V.O., Socher, R.: Non-autoregressive neural machine translation. In: ICLR (2018)
13. He, K., Girshick, R., Dollár, P.: Rethinking imagenet pre-training. In: ICCV (2019)
14. He, K., Gkioxari, G., Dollár, P., Girshick, R.B.: Mask R-CNN. In: ICCV (2017)
15. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)
16. Hosang, J.H., Benenson, R., Schiele, B.: Learning non-maximum suppression. In: CVPR (2017)
17. Hu, H., Gu, J., Zhang, Z., Dai, J., Wei, Y.: Relation networks for object detection. In: CVPR (2018)
18. Kirillov, A., Girshick, R., He, K., Dollár, P.: Panoptic feature pyramid networks. In: CVPR (2019)
19. Kirillov, A., He, K., Girshick, R., Rother, C., Dollar, P.: Panoptic segmentation. In: CVPR (2019)
20. Kuhn, H.W.: The hungarian method for the assignment problem (1955)
21. Li, Y., Qi, H., Dai, J., Ji, X., Wei, Y.: Fully convolutional instance-aware semantic segmentation. In: CVPR (2017)
22. Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: CVPR (2017)
23. Lin, T.Y., Goyal, P., Girshick, R.B., He, K., Dollár, P.: Focal loss for dense object detection. In: ICCV (2017)
24. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft COCO: Common objects in context. In: ECCV (2014)
25. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S.E., Fu, C.Y., Berg, A.C.: Ssd: Single shot multibox detector. In: ECCV (2016)
26. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. In: ICLR (2017)
27. Lüscher, C., Beck, E., Irie, K., Kitza, M., Michel, W., Zeyer, A., Schlüter, R., Ney, H.: Rwth asr systems for librispeech: Hybrid vs attention - w/o data augmentation. arXiv:1905.03072 (2019)
28. Milletari, F., Navab, N., Ahmadi, S.A.: V-net: Fully convolutional neural networks for volumetric medical image segmentation. In: 3DV (2016)
29. Oord, A.v.d., Li, Y., Babuschkin, I., Simonyan, K., Vinyals, O., Kavukcuoglu, K., Driessche, G.v.d., Lockhart, E., Cobo, L.C., Stimberg, F., et al.: Parallel wavenet: Fast high-fidelity speech synthesis. arXiv:1711.10433 (2017)
30. Park, E., Berg, A.C.: Learning to decompose for object detection and instance segmentation. arXiv:1511.06449 (2015)
31. Parmar, N., Vaswani, A., Uszkoreit, J., Kaiser, L., Shazeer, N., Ku, A., Tran, D.: Image transformer. In: ICML (2018)
32. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: Pytorch: An imperative style, high-performance deep learning library. In: NeurIPS (2019)
33. Pineda, L., Salvador, A., Drozdzal, M., Romero, A.: Elucidating image-to-set prediction: An analysis of models, losses and datasets. arXiv:1904.05709 (2019)
34. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I.: Language models are unsupervised multitask learners (2019)
35. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: CVPR (2016)
36. Ren, M., Zemel, R.S.: End-to-end instance segmentation with recurrent attention. In: CVPR (2017)

37. Ren, S., He, K., Girshick, R.B., Sun, J.: Faster R-CNN: Towards real-time object detection with region proposal networks. PAMI (2015)
38. Rezatofighi, H., Tsoi, N., Gwak, J., Sadeghian, A., Reid, I., Savarese, S.: Generalized intersection over union. In: CVPR (2019)
39. Rezatofighi, S.H., Kaskman, R., Motlagh, F.T., Shi, Q., Cremers, D., Leal-Taixé, L., Reid, I.: Deep perm-set net: Learn to predict sets with unknown permutation and cardinality using deep neural networks. arXiv:1805.00613 (2018)
40. Rezatofighi, S.H., Milan, A., Abbasnejad, E., Dick, A., Reid, I., Kaskman, R., Cremers, D., Leal-Taixé, L.: Deepsetnet: Predicting sets with deep neural networks. In: ICCV (2017)
41. Romera-Paredes, B., Torr, P.H.S.: Recurrent instance segmentation. In: ECCV (2015)
42. Salvador, A., Bellver, M., Baradad, M., Marqués, F., Torres, J., Giró, X.: Recurrent neural networks for semantic instance segmentation. arXiv:1712.00617 (2017)
43. Stewart, R.J., Andriluka, M., Ng, A.Y.: End-to-end people detection in crowded scenes. In: CVPR (2015)
44. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: NeurIPS (2014)
45. Synnaeve, G., Xu, Q., Kahn, J., Grave, E., Likhomanenko, T., Pratap, V., Srihari, A., Liptchinsky, V., Collobert, R.: End-to-end ASR: from supervised to semi-supervised learning with modern architectures. arXiv:1911.08460 (2019)
46. Tian, Z., Shen, C., Chen, H., He, T.: FCOS: Fully convolutional one-stage object detection. In: ICCV (2019)
47. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: NeurIPS (2017)
48. Vinyals, O., Bengio, S., Kudlur, M.: Order matters: Sequence to sequence for sets. In: ICLR (2016)
49. Wang, X., Girshick, R.B., Gupta, A., He, K.: Non-local neural networks. In: CVPR (2018)
50. Wu, Y., Kirillov, A., Massa, F., Lo, W.Y., Girshick, R.: Detectron2. <https://github.com/facebookresearch/detectron2> (2019)
51. Xiong, Y., Liao, R., Zhao, H., Hu, R., Bai, M., Yumer, E., Urtasun, R.: Upsnet: A unified panoptic segmentation network. In: CVPR (2019)
52. Zhang, S., Chi, C., Yao, Y., Lei, Z., Li, S.Z.: Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection. arXiv:1912.02424 (2019)
53. Zhou, X., Wang, D., Krähenbühl, P.: Objects as points. arXiv:1904.07850 (2019)

37. Ren, S., He, K., Girshick, R.B., Sun, J.: Faster R-CNN: Towards real-time object detection with region proposal networks. PAMI (2015)
38. Rezatofighi, H., Tsoi, N., Gwak, J., Sadeghian, A., Reid, I., Savarese, S.: Generalized intersection over union. In: CVPR (2019)
39. Rezatofighi, S.H., Kaskman, R., Motlagh, F.T., Shi, Q., Cremers, D., Leal-Taixé, L., Reid, I.: Deep perm-set net: Learn to predict sets with unknown permutation and cardinality using deep neural networks. arXiv:1805.00613 (2018)
40. Rezatofighi, S.H., Milan, A., Abbasnejad, E., Dick, A., Reid, I., Kaskman, R., Cremers, D., Leal-Taixé, L.: Deepsetnet: Predicting sets with deep neural networks. In: ICCV (2017)
41. Romera-Paredes, B., Torr, P.H.S.: Recurrent instance segmentation. In: ECCV (2015)
42. Salvador, A., Bellver, M., Baradad, M., Marqués, F., Torres, J., Giró, X.: Recurrent neural networks for semantic instance segmentation. arXiv:1712.00617 (2017)
43. Stewart, R.J., Andriluka, M., Ng, A.Y.: End-to-end people detection in crowded scenes. In: CVPR (2015)
44. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: NeurIPS (2014)
45. Synnaeve, G., Xu, Q., Kahn, J., Grave, E., Likhomanenko, T., Pratap, V., Sri-ram, A., Liptchinsky, V., Collobert, R.: End-to-end ASR: from supervised to semi-supervised learning with modern architectures. arXiv:1911.08460 (2019)
46. Tian, Z., Shen, C., Chen, H., He, T.: FCOS: Fully convolutional one-stage object detection. In: ICCV (2019)
47. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: NeurIPS (2017)
48. Vinyals, O., Bengio, S., Kudlur, M.: Order matters: Sequence to sequence for sets. In: ICLR (2016)
49. Wang, X., Girshick, R.B., Gupta, A., He, K.: Non-local neural networks. In: CVPR (2018)
50. Wu, Y., Kirillov, A., Massa, F., Lo, W.Y., Girshick, R.: Detectron2. <https://github.com/facebookresearch/detectron2> (2019)
51. Xiong, Y., Liao, R., Zhao, H., Hu, R., Bai, M., Yumer, E., Urtasun, R.: Upsnet: A unified panoptic segmentation network. In: CVPR (2019)
52. Zhang, S., Chi, C., Yao, Y., Lei, Z., Li, S.Z.: Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection. arXiv:1912.02424 (2019)
53. Zhou, X., Wang, D., Krähenbühl, P.: Objects as points. arXiv:1904.07850 (2019)

A Appendix

A.1 前提条件：マルチヘッド注意層

我々のモデルはTransformerアーキテクチャに基づいているので、ここで網羅性のために使用する注意メカニズムの一般的な形式を思い出す。注意のメカニズムは[47]に従うが、位置エンコーディングの詳細(式8参照)は[7]に従う。

マルチヘッド d 次元の M 個のヘッドを持つマルチヘッド注意の一般形は、以下のシグネチャを持つ関数である ($d^0 = \frac{d}{M}$ を使用し、アンダーブリースで行列/テンソルサイズを与える)。

$$\text{mh-attn} : \underbrace{X_q}_{d \times N_q}, \underbrace{X_{kv}}_{d \times N_{kv}}, \underbrace{T}_{M \times 3 \times d' \times d}, \underbrace{L}_{d \times d} \mapsto \underbrace{\tilde{X}_q}_{d \times N_q} \quad (3)$$

ここで、 X_q は長さ N_q のクエリ列、 X_{kv} は長さ N_{kv} のキー/値列(説明を簡単にするためにチャンネル数 d は同じ)、 T はいわゆるクエリ、キー、値の埋め込みを計算する重みテンソル、 L は射影行列である。出力はクエリシーケンスと同じサイズである。詳細を述べる前に語彙を固定するために、多頭自己注意(mh-s-attn)は $X_q = X_{kv}$ の特殊な場合である。

$$\text{mh-s-attn}(X, T, L) = \text{mh-attn}(X, X, T, L). \quad (4)$$

マルチヘッドアテンションは、単純に M 個のシングルアテンションヘッドを連結し、 L で投影するものである。一般的なプラクティス[47]は、残差接続、ドロップアウト、レイヤー正規化を使用することである。つまり、 $X_q = \text{mh-attn}(X_q, X_{kv}, T, L)$ 、 $X^{-(q)}$ を注意ヘッドの連結とすると、

$$X'_q = [\text{attn}(X_q, X_{kv}, T_1); \dots; \text{attn}(X_q, X_{kv}, T_M)] \quad (5)$$

$$\tilde{X}_q = \text{layernorm}(X_q + \text{dropout}(LX'_q)), \quad (6)$$

where $[;]$ denotes concatenation on the channel axis.

$\text{attn}(X_q, X_{kv}, T^0)$ で示される重みテンソル $T^0 \in \mathbb{R}^{3 \times d^0 \times d}$ を持つ注意ヘッドは、追加の位置エンコーディング $P_q \in \mathbb{R}^{d \times N_q}$ と $P_{kv} \in \mathbb{R}^{d \times N_{kv}}$ に依存する。クエリとキーの位置エンコーディングを追加した後、いわゆるクエリ、キー、値の埋め込みを計算することから始まる[7]：

$$[Q; K; V] = [T'_1(X_q + P_q); T'_2(X_{kv} + P_{kv}); T'_3 X_{kv}] \quad (7)$$

ここで、 T^0 は T_1^0, T_2^0, T_3^0 の連結である。次に、注目重み α は、クエリとキー間のドット積のソフトマックスに基づいて計算され、クエリシーケンスの各要素がキー値シーケンスのすべての要素に注目するようにする(i はクエリインデックス、 j はキー値インデックス)：

$$\alpha_{i,j} = \frac{e^{\frac{1}{\sqrt{d'}} Q_i^T K_j}}{Z_i} \quad \text{where } Z_i = \sum_{j=1}^{N_{kv}} e^{\frac{1}{\sqrt{d'}} Q_i^T K_j}. \quad (8)$$

A Appendix

A.1 Preliminaries: Multi-head attention layers

Since our model is based on the Transformer architecture, we remind here the general form of attention mechanisms we use for exhaustivity. The attention mechanism follows [47], except for the details of positional encodings (see Equation 8) that follows [7].

Multi-head The general form of *multi-head attention* with M heads of dimension d is a function with the following signature (using $d' = \frac{d}{M}$, and giving matrix/tensors sizes in underbrace)

$$\text{mh-attn} : \underbrace{X_q}_{d \times N_q}, \underbrace{X_{kv}}_{d \times N_{kv}}, \underbrace{T}_{M \times 3 \times d' \times d}, \underbrace{L}_{d \times d} \mapsto \underbrace{\tilde{X}_q}_{d \times N_q} \quad (3)$$

where X_q is the *query sequence* of length N_q , X_{kv} is the *key-value sequence* of length N_{kv} (with the same number of channels d for simplicity of exposition), T is the weight tensor to compute the so-called query, key and value embeddings, and L is a projection matrix. The output is the same size as the query sequence. To fix the vocabulary before giving details, multi-head *self*-attention (mh-s-attn) is the special case $X_q = X_{kv}$, i.e.

$$\text{mh-s-attn}(X, T, L) = \text{mh-attn}(X, X, T, L). \quad (4)$$

The multi-head attention is simply the concatenation of M single attention heads followed by a projection with L . The common practice [47] is to use residual connections, dropout and layer normalization. In other words, denoting $\tilde{X}_q = \text{mh-attn}(X_q, X_{kv}, T, L)$ and $\bar{X}^{(q)}$ the concatenation of attention heads, we have

$$X'_q = [\text{attn}(X_q, X_{kv}, T_1); \dots; \text{attn}(X_q, X_{kv}, T_M)] \quad (5)$$

$$\tilde{X}_q = \text{layernorm}(X_q + \text{dropout}(LX'_q)), \quad (6)$$

where $[;]$ denotes concatenation on the channel axis.

Single head An attention head with weight tensor $T' \in \mathbb{R}^{3 \times d' \times d}$, denoted by $\text{attn}(X_q, X_{kv}, T')$, depends on additional positional encoding $P_q \in \mathbb{R}^{d \times N_q}$ and $P_{kv} \in \mathbb{R}^{d \times N_{kv}}$. It starts by computing so-called query, key and value embeddings after adding the query and key positional encodings [7]:

$$[Q; K; V] = [T'_1(X_q + P_q); T'_2(X_{kv} + P_{kv}); T'_3 X_{kv}] \quad (7)$$

where T' is the concatenation of T'_1, T'_2, T'_3 . The *attention weights* α are then computed based on the softmax of dot products between queries and keys, so that each element of the query sequence attends to all elements of the key-value sequence (i is a query index and j a key-value index):

$$\alpha_{i,j} = \frac{e^{\frac{1}{\sqrt{d'}} Q_i^T K_j}}{Z_i} \quad \text{where } Z_i = \sum_{j=1}^{N_{kv}} e^{\frac{1}{\sqrt{d'}} Q_i^T K_j}. \quad (8)$$

我々の場合、位置エンコーディングは学習または固定かもしれないが、与えられたクエリ/キー値シーケンスに対して全ての注意層で共有しているので、注意のパラメータとして明示的に書かない。エンコーダとデコーダを記述する際の正確な値について、より詳細に説明する。最終的な出力は、注目重みで重み付けされた値の集約である： i 番目の行は $\text{attn}_i(X_q, X_{kv}, T^0) = \sum_{j=1}^{NkV} \alpha_{i,j} V_j$ で与えられる。

フィードフォワードネットワーク(FFN)層 オリジナルの変換器は、多頭注意層といわゆるFFN層[47]を交互に配置したもので、実質的に多層1x1畳み込みであり、我々の場合は M 個の入出力チャンネルを持つ。我々が考えるFFNは、ReLU活性化を持つ1x1畳み込みの2層で構成される。また、式6と同様に、2つの層の後に接続/ドロップアウト/レイヤーノームが残存している。

A.2 Losses

完全性を期すため、我々のアプローチで使用した損失を詳細に示す。すべての損失はバッチ内のオブジェクト数で正規化される。各GPUはサブバッチを受信するため、一般にサブバッチはGPU間でバランスがとれていないため、ローカルバッチ内のオブジェクト数で正規化するだけでは不十分である。代わりに、すべてのサブバッチのオブジェクトの総数で正規化することが重要である。

ボックス損失 [41, 36]と同様に、 b^{\wedge} の ℓ_1 損失とともに、Union上のIntersectionのソフトバージョンを損失に使用する：

$$\mathcal{L}_{\text{box}}(b_{\sigma(i)}, \hat{b}_i) = \lambda_{\text{iou}} \mathcal{L}_{\text{iou}}(b_{\sigma(i)}, \hat{b}_i) + \lambda_{\text{L1}} \|b_{\sigma(i)} - \hat{b}_i\|_1, \quad (9)$$

ここで、 $\lambda_{\text{iou}}, \lambda_{\text{L1}} \in R$ はハイパーパラメータ、 $\mathcal{L}_{\text{iou}}(-)$ は一般化IoU [38]：

$$\mathcal{L}_{\text{iou}}(b_{\sigma(i)}, \hat{b}_i) = 1 - \left(\frac{|b_{\sigma(i)} \cap \hat{b}_i|}{|b_{\sigma(i)} \cup \hat{b}_i|} - \frac{|B(b_{\sigma(i)}, \hat{b}_i) \setminus b_{\sigma(i)} \cup \hat{b}_i|}{|B(b_{\sigma(i)}, \hat{b}_i)|} \right). \quad (10)$$

$| \cdot |$ は「面積」を意味し、ボックス座標の和と交点はボックス自体の略記として使用される。結合または交差の面積は、 $b_{\sigma(i)}$ と b^{\wedge}_i の一次関数の min / max によって計算されるので、確率勾配に対して十分な損失が得られる。 $B(b_{\sigma(i)}, b^{\wedge}_i)$ は、 $b_{\sigma(i)}$, b^{\wedge}_i を含む最大のボックスを意味する(B を含む領域もボックス座標の線形関数の最小/最大に基づいて計算される)。

DICE/F-1損失 [28] DICE係数はIntersection over Unionと密接な関係がある。モデルの生のマスクロジット予測を \hat{m} 、バイナリターゲットマスクを m とすると、損失は次のように定義される。

$$\mathcal{L}_{\text{DICE}}(m, \hat{m}) = 1 - \frac{2m\sigma(\hat{m}) + 1}{\sigma(\hat{m}) + m + 1} \quad (11)$$

where σ is the sigmoid function. This loss is normalized by the number of objects.

In our case, the positional encodings may be learnt or fixed, but are shared across all attention layers for a given query/key-value sequence, so we do not explicitly write them as parameters of the attention. We give more details on their exact value when describing the encoder and the decoder. The final output is the aggregation of values weighted by attention weights: The i -th row is given by $\text{attn}_i(X_q, X_{\text{kv}}, T') = \sum_{j=1}^{N_{\text{kv}}} \alpha_{i,j} V_j$.

Feed-forward network (FFN) layers The original transformer alternates multi-head attention and so-called FFN layers [47], which are effectively multi-layer 1x1 convolutions, which have Md input and output channels in our case. The FFN we consider is composed of two-layers of 1x1 convolutions with ReLU activations. There is also a residual connection/dropout/layernorm after the two layers, similarly to equation 6.

A.2 Losses

For completeness, we present in detail the losses used in our approach. All losses are normalized by the number of objects inside the batch. Extra care must be taken for distributed training: since each GPU receives a sub-batch, it is not sufficient to normalize by the number of objects in the local batch, since in general the sub-batches are not balanced across GPUs. Instead, it is important to normalize by the total number of objects in all sub-batches.

Box loss Similarly to [41,36], we use a soft version of Intersection over Union in our loss, together with a ℓ_1 loss on \hat{b} :

$$\mathcal{L}_{\text{box}}(b_{\sigma(i)}, \hat{b}_i) = \lambda_{\text{iou}} \mathcal{L}_{\text{iou}}(b_{\sigma(i)}, \hat{b}_i) + \lambda_{\text{L1}} \|b_{\sigma(i)} - \hat{b}_i\|_1, \quad (9)$$

where $\lambda_{\text{iou}}, \lambda_{\text{L1}} \in \mathbb{R}$ are hyperparameters and $\mathcal{L}_{\text{iou}}(\cdot)$ is the generalized IoU [38]:

$$\mathcal{L}_{\text{iou}}(b_{\sigma(i)}, \hat{b}_i) = 1 - \left(\frac{|b_{\sigma(i)} \cap \hat{b}_i|}{|b_{\sigma(i)} \cup \hat{b}_i|} - \frac{|B(b_{\sigma(i)}, \hat{b}_i) \setminus b_{\sigma(i)} \cup \hat{b}_i|}{|B(b_{\sigma(i)}, \hat{b}_i)|} \right). \quad (10)$$

$| \cdot |$ means “area”, and the union and intersection of box coordinates are used as shorthands for the boxes themselves. The areas of unions or intersections are computed by min / max of the linear functions of $b_{\sigma(i)}$ and \hat{b}_i , which makes the loss sufficiently well-behaved for stochastic gradients. $B(b_{\sigma(i)}, \hat{b}_i)$ means the largest box containing $b_{\sigma(i)}, \hat{b}_i$ (the areas involving B are also computed based on min / max of linear functions of the box coordinates).

DICE/F-1 loss [28] The DICE coefficient is closely related to the Intersection over Union. If we denote by \hat{m} the raw mask logits prediction of the model, and m the binary target mask, the loss is defined as:

$$\mathcal{L}_{\text{DICE}}(m, \hat{m}) = 1 - \frac{2m\sigma(\hat{m}) + 1}{\sigma(\hat{m}) + m + 1} \quad (11)$$

where σ is the sigmoid function. This loss is normalized by the number of objects.

A.3 詳細アーキテクチャ

DETRで使用される変換器の詳細な説明は、各注意層で渡される位置エンコーディングを図10に示す。CNNバックボーンからの画像特徴は、変換エンコーダを通過し、空間位置エンコーディングとともに、各マルチヘッド自己注意層でクエリとキーに追加される。次に、デコーダはクエリ(最初はゼロに設定)、位置エンコーディング(オブジェクトクエリ)、エンコーダメモリを受け取り、複数のマルチヘッド自己注意とデコーダエンコーダ注意によって、予測クラスラベルとバウンディングボックスの最終セットを生成する。最初のデコーダ層の最初の自己アテンション層はスキップできる。

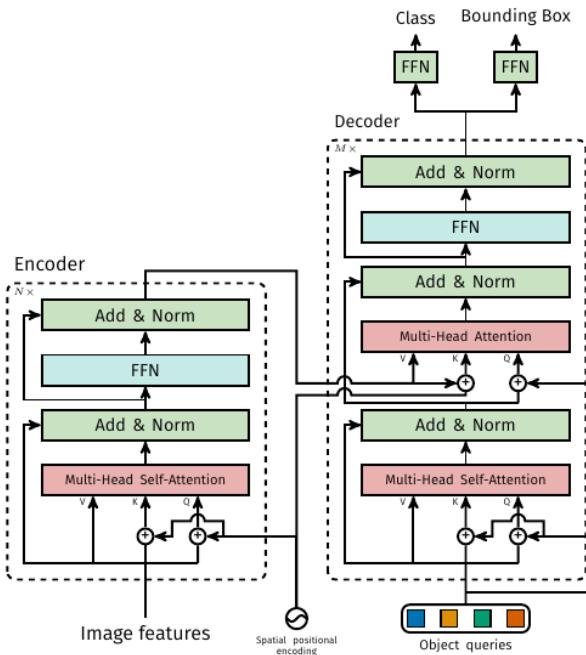


図10:DETRのトランスフォーマーのアーキテクチャ。詳細はセクションA.3を参照のこと。

計算複雑度エンコーダ内の全ての自己注意は複雑度 $O(d^2 \cdot HW + d \cdot (HW)^2)$ である： $O(d^0 \cdot d)$ は単一のクエリ/キー/値の埋め込み(と $M \cdot d^0 = d$)を計算するコストであり、 $O(d^0 \cdot (HW)^2)$ は1つのヘッドに対する注目重みを計算するコストである。その他の計算は無視できる。デコーダでは、各自己アテンションは $O(d^2 \cdot N + d \cdot N^2)$ であり、エンコーダとデコーダのクロスアテンションは $O(d^2 \cdot (N + HW) + d \cdot N \cdot HW)$ であり、実際には $N \cdot HW$ なのでエンコーダよりはるかに低い。

A.3 Detailed architecture

The detailed description of the transformer used in DETR, with positional encodings passed at every attention layer, is given in Fig. 10. Image features from the CNN backbone are passed through the transformer encoder, together with spatial positional encoding that are added to queries and keys at every multi-head self-attention layer. Then, the decoder receives queries (initially set to zero), output positional encoding (object queries), and encoder memory, and produces the final set of predicted class labels and bounding boxes through multiple multi-head self-attention and decoder-encoder attention. The first self-attention layer in the first decoder layer can be skipped.

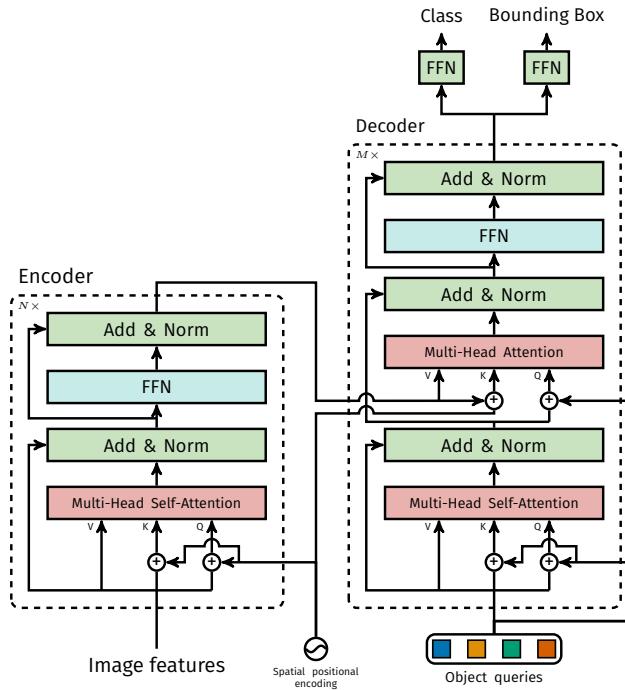


Fig. 10: Architecture of DETR’s transformer. Please, see Section A.3 for details.

Computational complexity Every self-attention in the encoder has complexity $\mathcal{O}(d^2HW + d(HW)^2)$: $\mathcal{O}(d'd)$ is the cost of computing a single query/key/value embeddings (and $Md' = d$), while $\mathcal{O}(d'(HW)^2)$ is the cost of computing the attention weights for one head. Other computations are negligible. In the decoder, each self-attention is in $\mathcal{O}(d^2N + dN^2)$, and cross-attention between encoder and decoder is in $\mathcal{O}(d^2(N + HW) + dNHW)$, which is much lower than the encoder since $N \ll HW$ in practice.

FLOPS計算 Faster R-CNNのFLOPSは画像内のプロポーザルの数に依存することを考慮し、COCO 2017検証セットの最初の100画像の平均FLOPS数を報告する。Detectron2[50]のツールフロップ数演算子を用いてFLOPSを計算する。Detectron2モデルにはそのまま使用し、DETRモデルにはバッチ行列乗算(bmm)を考慮するように拡張する。

A.4 学習ハイパーパラメータ

AdamW[26]を用いてDETRを学習し、 10^{-4} に設定した。また、勾配クリッピングを適用し、最大勾配ノルムを0.1とする。バックボーンとトランسفォーマーは若干異なる扱いをしているが、ここでは両者の詳細について述べる。

バックボーンImageNetの事前学習済みバックボーンResNet-50はTorchvisionからインポートされ、最後の分類層は破棄される。バックボーンバッチの正規化の重みと統計量は、物体検出で広く採用されている手法に従い、学習中に凍結される。学習率 10^{-5} でバックボーンを微調整する。バックボーンの学習率をネットワークの他の部分よりおよそ1桁小さくすることは、特に最初の数エポックにおいて、学習を安定させるために重要であることが観察される。

トランسفォーマー 10^{-4} の学習率でトランسفォーマーを学習する。レイヤーの正規化の前に、マルチヘッドアテンションとFFNの後に0.1の加算ドロップアウトが適用される。重みはXavier初期化でランダムに初期化される。

損失 $\lambda_{L1} = 5$ と $\lambda_{iou} = 2$ の重みで、バウンディングボックス回帰に ℓ_1 と GIoU損失の線形結合を用いる。すべてのモデルは $N = 100$ デコーダのクエリースロットで学習された。

ベースライン Faster-RCNN+のベースラインは、GIoU[38]損失とバウンディングボックス回帰の標準的な ℓ_1 損失を使用している。我々は、損失に対する最適な重みを見つけるためにグリッド探索を行い、最終的なモデルは、ボックス回帰タスクと提案回帰タスクにそれぞれ重み20と1を持つGIoU損失のみを使用する。ベースラインにはDETRで使用したものと同じデータ補強を採用し、9×スケジュール(約109エポック)で学習させる。他の設定はすべてDetectron2モデル動物園[50]の同じモデルと同じである。

空間位置エンコーディングエンコーダの活性化は、画像特徴の対応する空間位置と関連付けられる。我々のモデルでは、これらの空間位置を表現するために固定絶対エンコーディングを使用する。我々は、オリジナルのTransformer [47]エンコーディングを2Dの場合[31]に一般化したものを探用する。具体的には、各埋め込みの両空間座標に対して、周波数の異なる d_2 正弦関数と余弦関数を独立に使用する。次に、これらを連結して、最終的な d チャンネルの位置エンコーディングを得る。

A.5 その他の結果

DETR-R101モデルのパノプティック予測に関するいくつかの追加定性的結果をFig. 11に示す。

FLOPS computation Given that the FLOPS for Faster R-CNN depends on the number of proposals in the image, we report the average number of FLOPS for the first 100 images in the COCO 2017 validation set. We compute the FLOPS with the tool `flop_count_operators` from Detectron2 [50]. We use it without modifications for Detectron2 models, and extend it to take batch matrix multiply (`bmm`) into account for DETR models.

A.4 Training hyperparameters

We train DETR using AdamW [26] with improved weight decay handling, set to 10^{-4} . We also apply gradient clipping, with a maximal gradient norm of 0.1. The backbone and the transformers are treated slightly differently, we now discuss the details for both.

Backbone ImageNet pretrained backbone ResNet-50 is imported from Torchvision, discarding the last classification layer. Backbone batch normalization weights and statistics are frozen during training, following widely adopted practice in object detection. We fine-tune the backbone using learning rate of 10^{-5} . We observe that having the backbone learning rate roughly an order of magnitude smaller than the rest of the network is important to stabilize training, especially in the first few epochs.

Transformer We train the transformer with a learning rate of 10^{-4} . Additive dropout of 0.1 is applied after every multi-head attention and FFN before layer normalization. The weights are randomly initialized with Xavier initialization.

Losses We use linear combination of ℓ_1 and GIoU losses for bounding box regression with $\lambda_{\text{L1}} = 5$ and $\lambda_{\text{iou}} = 2$ weights respectively. All models were trained with $N = 100$ decoder query slots.

Baseline Our enhanced Faster-RCNN+ baselines use GIoU [38] loss along with the standard ℓ_1 loss for bounding box regression. We performed a grid search to find the best weights for the losses and the final models use only GIoU loss with weights 20 and 1 for box and proposal regression tasks respectively. For the baselines we adopt the same data augmentation as used in DETR and train it with $9\times$ schedule (approximately 109 epochs). All other settings are identical to the same models in the Detectron2 model zoo [50].

Spatial positional encoding Encoder activations are associated with corresponding spatial positions of image features. In our model we use a fixed absolute encoding to represent these spatial positions. We adopt a generalization of the original Transformer [47] encoding to the 2D case [31]. Specifically, for both spatial coordinates of each embedding we independently use $\frac{d}{2}$ sine and cosine functions with different frequencies. We then concatenate them to get the final d channel positional encoding.

A.5 Additional results

Some extra qualitative results for the panoptic prediction of the DETR-R101 model are shown in Fig.11.



(a) オブジェクトが重なった場合の失敗例。PanopticFPNは1つの平面を完全に見逃してしまい、DETRはそのうちの3つを正確にセグメンテーションできていない。



(b) 物マスクはフル解像度で予測されるため、PanopticFPNよりもシャープな境界が可能である。

図11:パノプティック予測の比較。左から右へ: グラウンドトゥルース、ResNet 101を用いたPanopticFPN、ResNet 101を用いたDETR

インスタンス数の増加 DETRは設計上、クエリスロットを持つよりも多くのオブジェクトを予測できない、つまり我々の実験では100個である。本節では、この限界に近づいたときのDETRの挙動を分析する。与えられたクラスの正準正方形画像を選択し、 10×10 グリッドでこれを繰り返し、モデルによって見逃されたインスタンスのパーセンテージを計算する。100インスタンス未満でモデルをテストするために、いくつかのセルをランダムにマスクする。これにより、何個見えてもオブジェクトの絶対的な大きさが同じであることが保証される。マスキングのランダム性を考慮し、異なるマスクで100回実験を繰り返した。結果をFig. 12に示す。クラス間で挙動が似ており、最大50個が見えると全てのインスタンスを検出するが、その後飽和を開始し、より多くのインスタンスを見逃す。注目すべきは、画像に100個のインスタンスがすべて含まれている場合、モデルは平均して30個しか検出しないが、これは画像に50個のインスタンスしか含まれていない場合よりも少ないことである。このモデルの直感に反する挙動は、画像と検出が学習分布からかけ離れているためと思われる。

このテストは、1つのクラスのインスタンスが多数存在する例画像が非常に少ないため、設計上、分布外での汎化のテストであることに注意。実験から、領域外汎化の2つのタイプ、すなわち、画像そのものとクラスごとのオブジェクトの数を分離することは困難である。しかし、COCO画像には同じクラスのオブジェクトが多数含まれているものはほとんどないため、この種の実験は、クエリオブジェクトがデータセットのラベルと位置の分布に過剰適合しているかどうかを理解するための我々の最善の努力である。全体として、このモデルは50オブジェクトまで完璧に近い検出結果を得ることができるために、これらの分布にオーバーフィットしないことが実験から示唆された。



(a) Failure case with overlapping objects. PanopticFPN misses one plane entirely, while DETR fails to accurately segment 3 of them.



(b) `Things` masks are predicted at full resolution, which allows sharper boundaries than PanopticFPN

Fig. 11: Comparison of panoptic predictions. From left to right: Ground truth, PanopticFPN with ResNet 101, DETR with ResNet 101

Increasing the number of instances By design, DETR cannot predict more objects than it has query slots, i.e. 100 in our experiments. In this section, we analyze the behavior of DETR when approaching this limit. We select a canonical square image of a given class, repeat it on a 10×10 grid, and compute the percentage of instances that are missed by the model. To test the model with less than 100 instances, we randomly mask some of the cells. This ensures that the absolute size of the objects is the same no matter how many are visible. To account for the randomness in the masking, we repeat the experiment 100 times with different masks. The results are shown in Fig. 12. The behavior is similar across classes, and while the model detects all instances when up to 50 are visible, it then starts saturating and misses more and more instances. Notably, when the image contains all 100 instances, the model only detects 30 on average, which is less than if the image contains only 50 instances that are all detected. The counter-intuitive behavior of the model is likely because the images and the detections are far from the training distribution.

Note that this test is a test of generalization out-of-distribution by design, since there are very few example images with a lot of instances of a single class. It is difficult to disentangle, from the experiment, two types of out-of-domain generalization: the image itself vs the number of object per class. But since few to no COCO images contain only a lot of objects of the same class, this type of experiment represents our best effort to understand whether query objects overfit the label and position distribution of the dataset. Overall, the experiments suggests that the model does not overfit on these distributions since it yields near-perfect detections up to 50 objects.

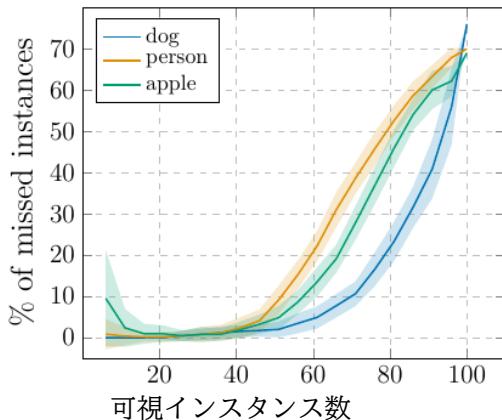


図12:DETRが見逃した様々なクラスのインスタンス数の、画像に存在する数による分析。平均値と標準偏差を報告する。インスタンス数が100に近づくと、DETRは飽和し始め、より多くのオブジェクトを見逃すようになる

A.6 PyTorch推論コード

アプローチの単純さを示すために、PyTorchとTorchvisionライブラリを使った推論コードをリスト1に含める。コードはPython 3.6+、PyTorch 1.4、Torchvision 0.5で実行される。バッチングをサポートしていないため、GPUごとに1つの画像を持つDistributedDataParallelでの推論やトレーニングにのみ適していることに注意。また、このコードではわかりやすくするために、エンコーダで固定の代わりに学習された位置エンコーディングを使用し、位置エンコーディングは各変換層ではなく、入力にのみ追加されることに注意してください。これらの変更には、PyTorchによる変換器の実装を超える必要があり、可読性に支障をきたす。実験を再現するための全コードは、会議の前に公開される予定である。

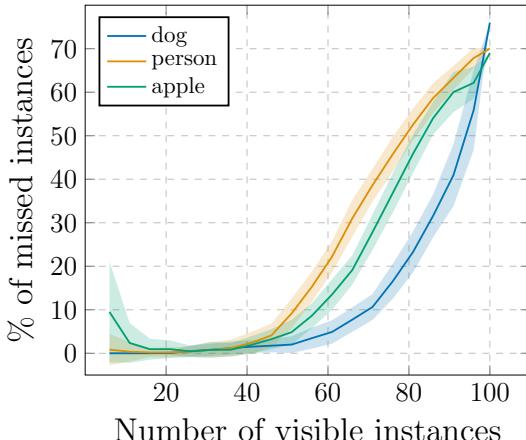


Fig. 12: Analysis of the number of instances of various classes missed by DETR depending on how many are present in the image. We report the mean and the standard deviation. As the number of instances gets close to 100, DETR starts saturating and misses more and more objects

A.6 PyTorch inference code

To demonstrate the simplicity of the approach, we include inference code with PyTorch and Torchvision libraries in Listing 1. The code runs with Python 3.6+, PyTorch 1.4 and Torchvision 0.5. Note that it does not support batching, hence it is suitable only for inference or training with DistributedDataParallel with one image per GPU. Also note that for clarity, this code uses learnt positional encodings in the encoder instead of fixed, and positional encodings are added to the input only instead of at each transformer layer. Making these changes requires going beyond PyTorch implementation of transformers, which hampers readability. The entire code to reproduce the experiments will be made available before the conference.

```

1 import torch
2 from torch import nn
3 from torchvision.models import resnet50
4
5 class DETR(nn.Module):
6
7     def __init__(self, num_classes, hidden_dim, nheads,
8                  num_encoder_layers, num_decoder_layers):
9         super().__init__()
10        # We take only convolutional layers from ResNet-50 model
11        self.backbone = nn.Sequential(*list(resnet50(pretrained=True).children())[:-2])
12        self.conv = nn.Conv2d(2048, hidden_dim, 1)
13        self.transformer = nn.Transformer(hidden_dim, nheads,
14                                         num_encoder_layers, num_decoder_layers)
15        self.linear_class = nn.Linear(hidden_dim, num_classes + 1)
16        self.linear_bbox = nn.Linear(hidden_dim, 4)
17        self.query_pos = nn.Parameter(torch.rand(100, hidden_dim))
18        self.row_embed = nn.Parameter(torch.rand(50, hidden_dim // 2))
19        self.col_embed = nn.Parameter(torch.rand(50, hidden_dim // 2))
20
21    def forward(self, inputs):
22        x = self.backbone(inputs)
23        h = self.conv(x)
24        H, W = h.shape[-2:]
25        pos = torch.cat([
26            self.col_embed[:W].unsqueeze(0).repeat(H, 1, 1),
27            self.row_embed[:H].unsqueeze(1).repeat(1, W, 1),
28        ], dim=-1).flatten(0, 1).unsqueeze(1)
29        h = self.transformer(pos + h.flatten(2).permute(2, 0, 1),
30                            self.query_pos.unsqueeze(1))
31        return self.linear_class(h), self.linear_bbox(h).sigmoid()
32
33 detr = DETR(num_classes=91, hidden_dim=256, nheads=8, num_encoder_layers=6, num_decoder_layers=6)
34 detr.eval()
35 inputs = torch.randn(1, 3, 800, 1200)
36 logits, bboxes = detr(inputs)

```

リスト1: DETR PyTorch推論コード。わかりやすくするために、エンコーダでは固定の代わりに学習された位置エンコーディングを使用し、位置エンコーディングは各変換層ではなく入力のみに追加される。これらの変更には、PyTorchによる変換器の実装を超える必要があり、可読性に支障をきたす。実験を再現するための全コードは、会議の前に公開される予定である。

```

1 import torch
2 from torch import nn
3 from torchvision.models import resnet50
4
5 class DETR(nn.Module):
6
7     def __init__(self, num_classes, hidden_dim, nheads,
8                  num_encoder_layers, num_decoder_layers):
9         super().__init__()
10        # We take only convolutional layers from ResNet-50 model
11        self.backbone = nn.Sequential(*list(resnet50(pretrained=True).children())[:-2])
12        self.conv = nn.Conv2d(2048, hidden_dim, 1)
13        self.transformer = nn.Transformer(hidden_dim, nheads,
14                                         num_encoder_layers, num_decoder_layers)
15        self.linear_class = nn.Linear(hidden_dim, num_classes + 1)
16        self.linear_bbox = nn.Linear(hidden_dim, 4)
17        self.query_pos = nn.Parameter(torch.rand(100, hidden_dim))
18        self.row_embed = nn.Parameter(torch.rand(50, hidden_dim // 2))
19        self.col_embed = nn.Parameter(torch.rand(50, hidden_dim // 2))
20
21    def forward(self, inputs):
22        x = self.backbone(inputs)
23        h = self.conv(x)
24        H, W = h.shape[-2:]
25        pos = torch.cat([
26            self.col_embed[:W].unsqueeze(0).repeat(H, 1, 1),
27            self.row_embed[:H].unsqueeze(1).repeat(1, W, 1),
28        ], dim=-1).flatten(0, 1).unsqueeze(1)
29        h = self.transformer(pos + h.flatten(2).permute(2, 0, 1),
30                            self.query_pos.unsqueeze(1))
31        return self.linear_class(h), self.linear_bbox(h).sigmoid()
32
33 detr = DETR(num_classes=91, hidden_dim=256, nheads=8, num_encoder_layers=6, num_decoder_layers=6)
34 detr.eval()
35 inputs = torch.randn(1, 3, 800, 1200)
36 logits, bboxes = detr(inputs)

```

Listing 1: DETR PyTorch inference code. For clarity it uses learnt positional encodings in the encoder instead of fixed, and positional encodings are added to the input only instead of at each transformer layer. Making these changes requires going beyond PyTorch implementation of transformers, which hampers readability. The entire code to reproduce the experiments will be made available before the conference.